Title: Lecture 5: Skydiving algorithm

Speakers: Aike Liu

Collection: Mini-Course of Numerical Conformal Bootstrap

Date: April 28, 2023 - 10:00 AM

URL: https://pirsa.org/23040148

Skydiving Algorithm: A Dynamical SDP Solver

Aike Liu

Numerical Bootstrap Mini-Course: Lecture 5

Caltech SIMONS FOUNDATION PERIMETER INSTITUTE

Table of Contents

Lagrangian perspective of Numerical Bootstrap

$$L_\mu(x, y, X, Y) = c^T x + b^T y - x^T B y + \mathrm{Tr}\left((X - x^T A_*) Y\right) - \mu \log\det X$$

The stationarity equations of this Lagrangian with respect to x, y and Y yield exactly the primal and dual feasibility conditions.

$$c - By - \mathrm{Tr}\,(A_* Y) = 0, Y \geq 0$$
$$b - B^T x = 0, \quad X - x^T A_* = 0, X \geq 0$$

Demanding stationarity with respect to X yields:

$$XY = \mu I, \quad \mu \to 0, \quad \text{optimal.}$$

Lagrangian perspective of Numerical Bootstrap

$$c - By - \mathrm{Tr}\,(A_* Y) = 0,$$
$$b - B^T x = 0, \quad X - x^T A_* = 0,$$
$$XY = \mu I, \quad \mu \to 0.$$

At optimality,

$$\lim_{\mu \to 0} L_\mu(x^*, y^*, X^*, Y^*) = c^T x \approx b^T y$$

Define `duality gap` as

$$gap = c^T x - b^T y, \quad \lim_{\mu \to 0} gap = 0.$$

For convenience, we will write $\xi = (x, y, X, Y)$. Then equations above can be rewritten as

$$\lim_{\mu \to 0} \nabla_\xi L_\mu = 0$$

Interior Point Method μ

Original optimum condition,

$$XY = 0, \quad X \geq 0, Y \geq 0$$

The $(-\mu \log\det X)$ is a penalty term for $X \to 0$.

$$XY = \mu I, \quad X \geq 0, Y \geq 0$$

Gradually decrease μ by $\mu_{next} = \beta \mu_{curr}$

$$(X + dX)(Y + dY) = \mu_{next} I, \quad X + dX \geq 0, Y + dY \geq 0$$

- Force X and Y to stay positive.
- Condition X and Y to be regular.

Lagrangian perspective of Numerical Bootstrap

$$c - By - \mathrm{Tr}\,(A_* Y) = 0,$$
$$b - B^T x = 0, \quad X - x^T A_* = 0,$$
$$XY = \mu I, \quad \mu \to 0.$$

For convenience, we will write $\xi = (x, y, X, Y)$. Then equations above can be rewritten as

$$\lim_{\mu \to 0} \nabla_\xi L_\mu = 0$$

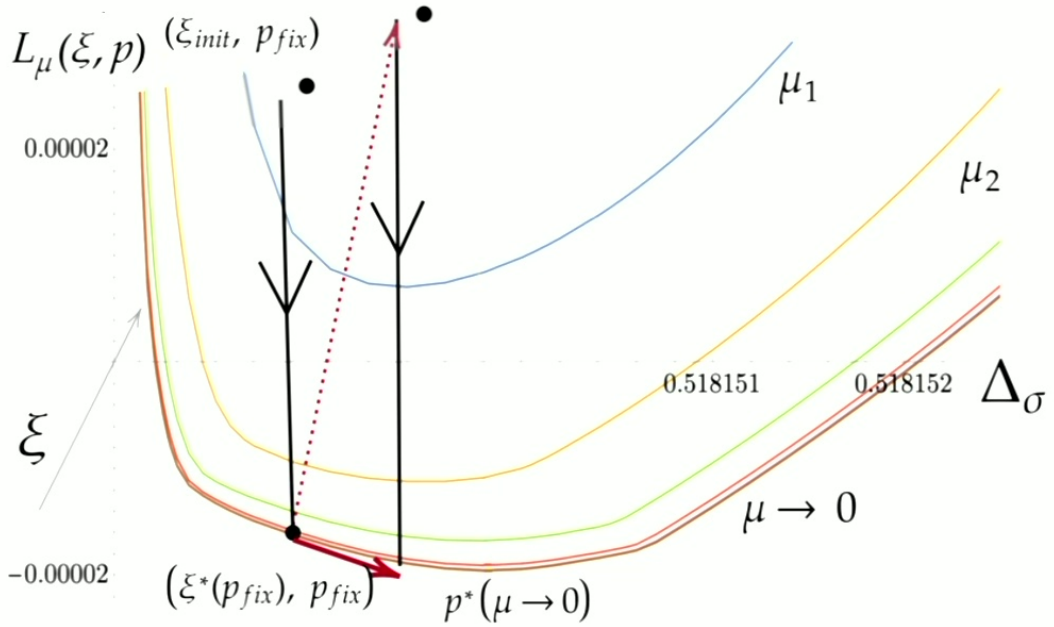
Physical parameters from bootstraps, denoted as p. For example,

$$p = (\Delta_\sigma, \Delta_\epsilon, \lambda_{\sigma\sigma\epsilon}/\lambda_{\epsilon\epsilon\epsilon})$$

Skydiving Algorithm: A Dynamical SDP Solver

Warm-up

SDPB

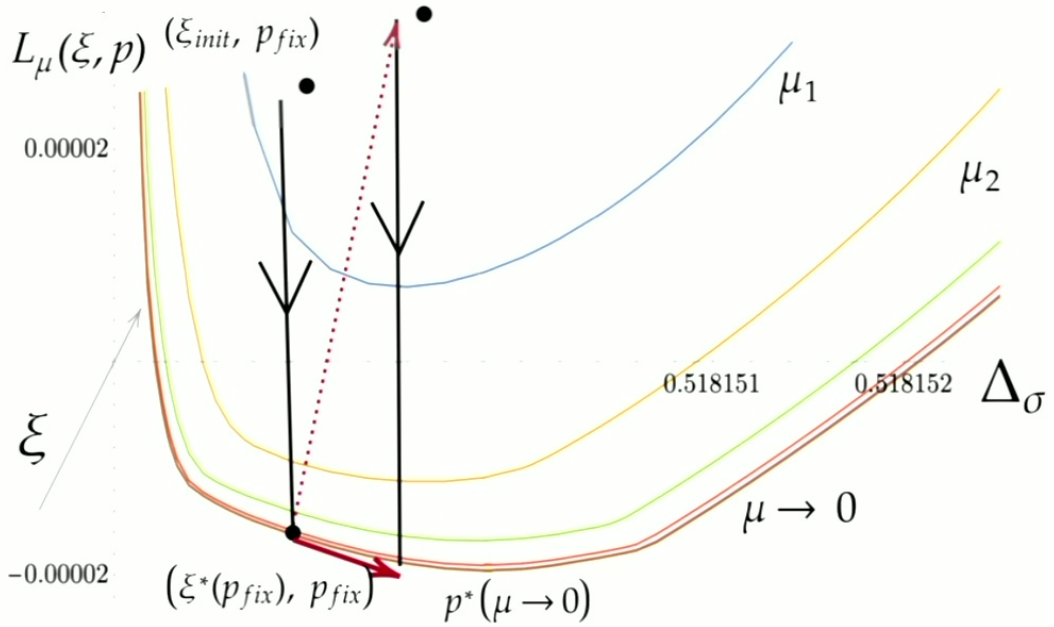


$$L_\mu(\xi, p_{\text{fixed}}) \to L_{\mu=0}(\xi^*, p_{\text{fixed}})$$
$$\to p'_{\text{fixed}} = p_{\text{fixed}} + \delta p$$
$$\to L_\mu(\xi, p'_{\text{fixed}}) \to L_{\mu=0}(\xi^*, p'_{\text{fixed}})$$

Skydiving Algorithm: A Dynamical SDP Solver

Warm-up

SDPB



$$L_\mu(\xi, p_{\text{fixed}}) \to L_{\mu=0}(\xi^*, p_{\text{fixed}})$$
$$\to p'_{\text{fixed}} = p_{\text{fixed}} + \delta p$$
$$\to L_\mu(\xi, p'_{\text{fixed}}) \to L_{\mu=0}(\xi^*, p'_{\text{fixed}})$$

Enlarged Solver

Costs of full or partial SDPB runs grow as $\dim(p)$ grows.
Can we explore the structures of the entire space at the same time?

Constraints depend on the CFT data p,

$$b(p), c(p), B(p)$$

Our optimization problem becomes

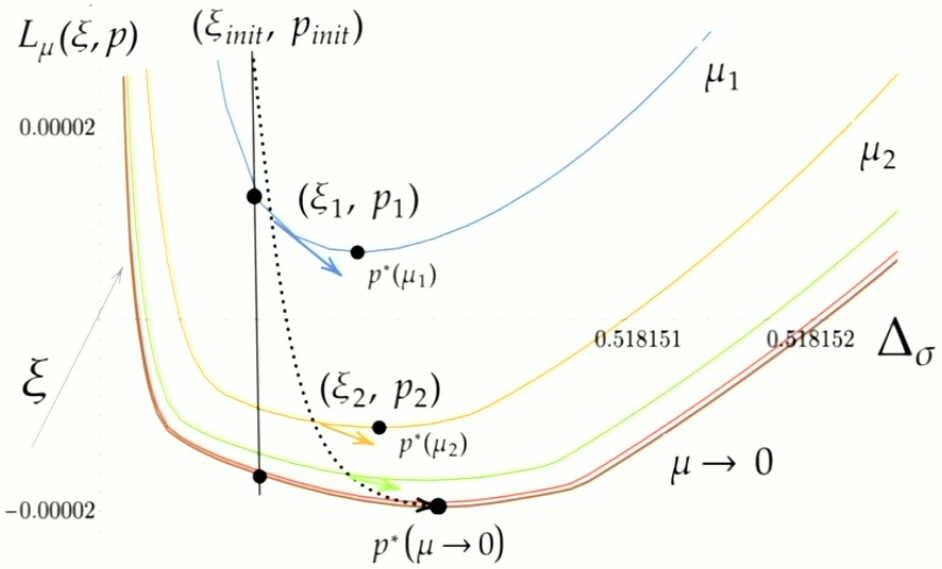
$$L(\xi) \to L(\xi, p),$$

$$\nabla_p L_\mu = 0 \to \nabla_p L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0,$$

where each iteration we take a step in the larger space,

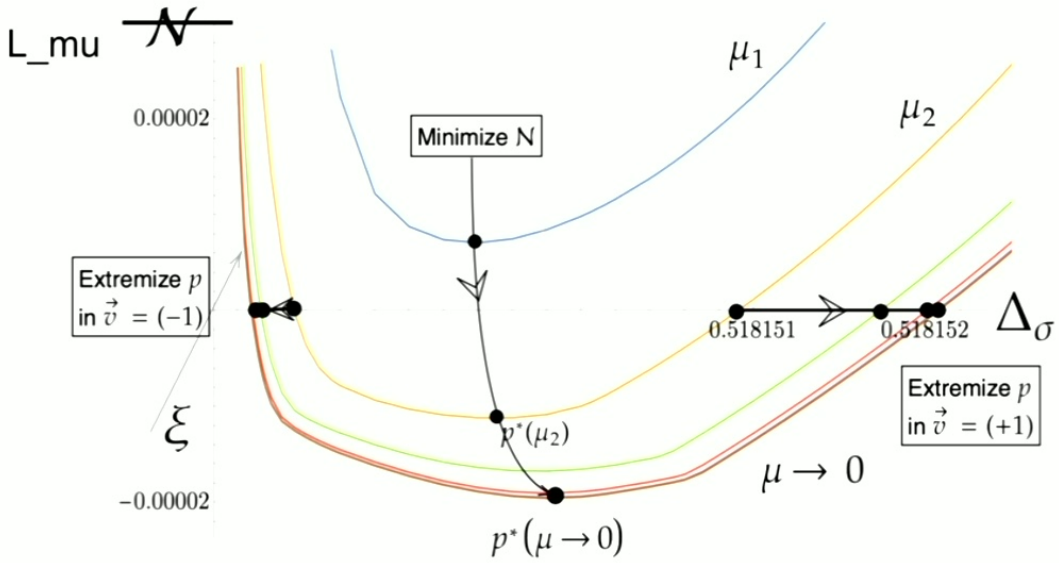
$$d\xi \to (d\xi, dp).$$

Goals



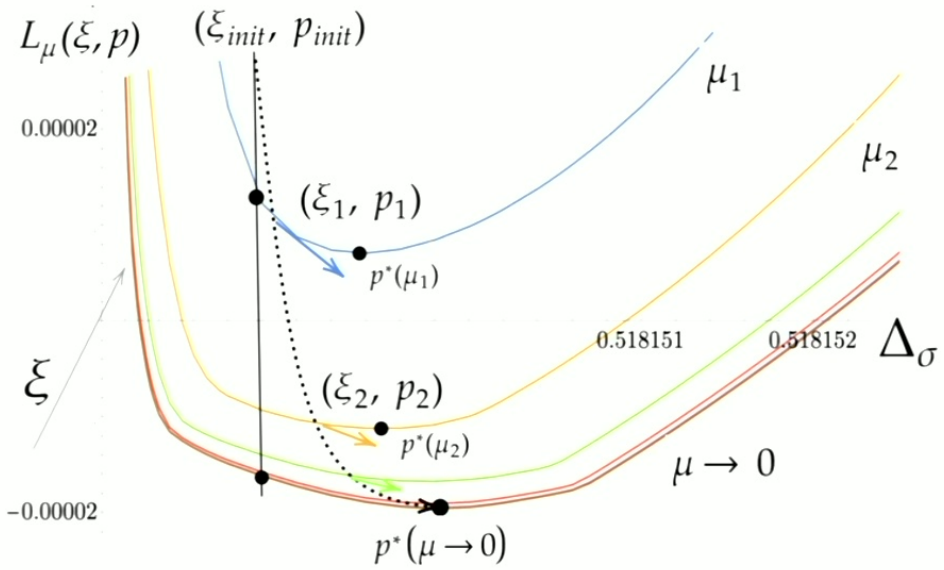
$$L_\mu(\xi, p) \to (d\xi, dp),\ \mu' < \mu$$
$$\to \ldots \to L_{\mu=0}(\xi^*(\mu = 0), p^*(\mu = 0))$$

Goals



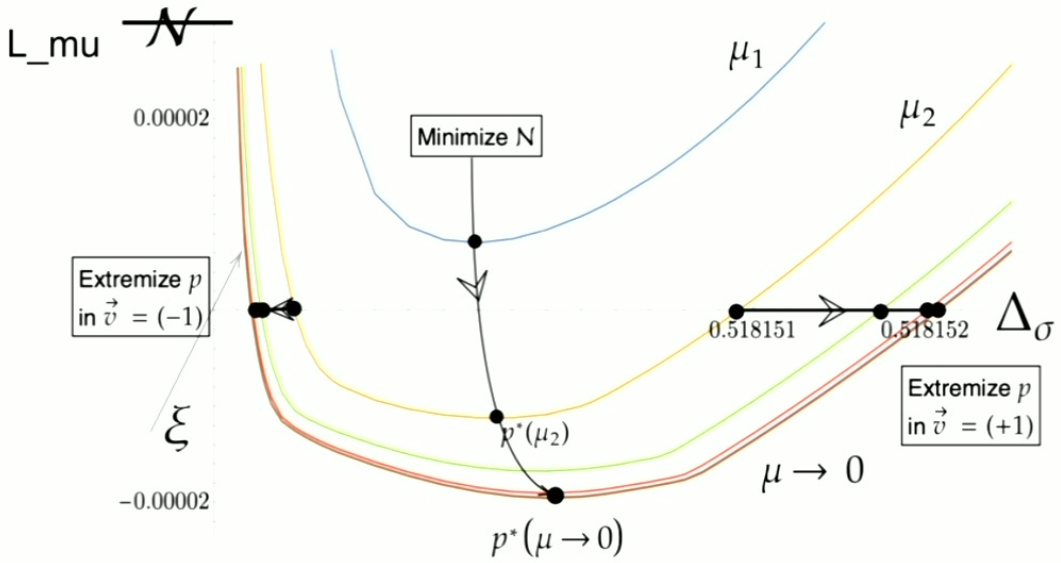
- Minimize $\mathcal{N}(p)$.
- Extremize p in a direction $\vec{v}$.

Goals



$$L_\mu(\xi, p) \to (d\xi, dp),\ \mu' < \mu$$
$$\to \ldots \to L_{\mu=0}(\xi^*(\mu = 0), p^*(\mu = 0))$$

Goals



- Minimize $\mathcal{N}(p)$.
- Extremize p in a direction $\vec{v}$.

Minimize $\mathcal{N}(p)$

- Stationary conditions

$$\nabla_p L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0.$$

- Newton update equations

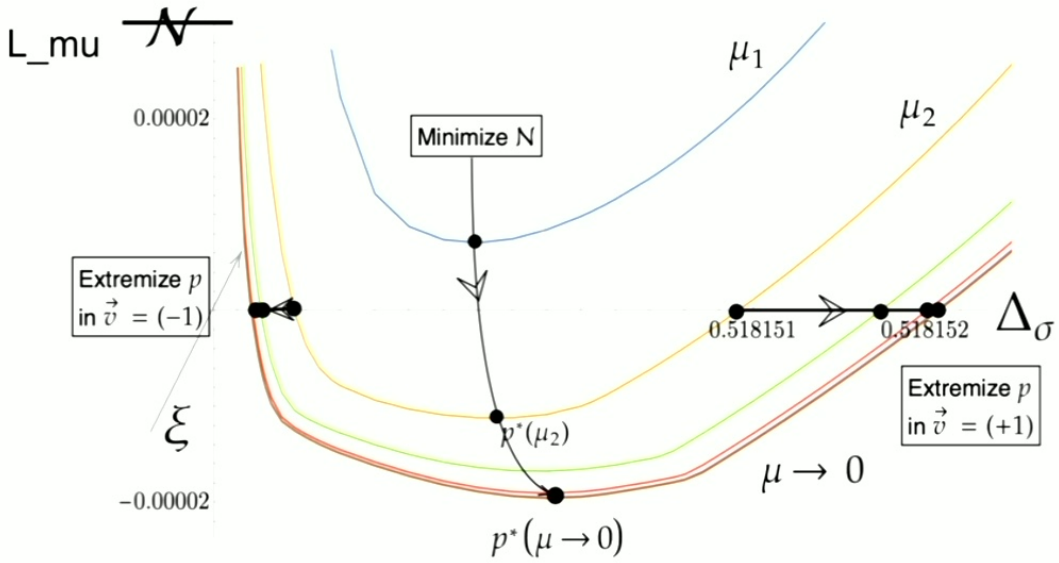
$$H_{pp}\delta p + H_{p\xi}\delta\xi = -\nabla_p L_\mu,$$
$$H_{\xi p}\delta p + H_{\xi\xi}\delta\xi = -\nabla_\xi L_\mu.$$

- Solve for Newton step

$$H^{\text{eff}}_{pp} = H_{pp} - H_{p\xi}H^{-1}_{\xi\xi}H_{\xi p},$$
$$\nabla^{\text{eff}}_p L_\mu = \nabla_p L_\mu - H_{p\xi}H^{-1}_{\xi\xi}\nabla_\xi L_\mu.$$

$$H^{\text{eff}}_{pp}\,\delta p = -\nabla^{\text{eff}}_p L_\mu,$$
$$H_{\xi\xi}\delta\xi = -\nabla_\xi L_\mu - H_{\xi p}\delta p.$$

Goals



- Minimize $\mathcal{N}(p)$.
- Extremize p in a direction $\vec{v}$.

Minimize $\mathcal{N}(p)$

- Stationary conditions

$$\nabla_p L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0.$$

- Newton update equations

$$H_{pp}\delta p + H_{p\xi}\delta\xi = -\nabla_p L_\mu,$$
$$H_{\xi p}\delta p + H_{\xi\xi}\delta\xi = -\nabla_\xi L_\mu.$$

- Solve for Newton step

$$H^{\text{eff}}_{pp} = H_{pp} - H_{p\xi}H^{-1}_{\xi\xi}H_{\xi p},$$
$$\nabla^{\text{eff}}_p L_\mu = \nabla_p L_\mu - H_{p\xi}H^{-1}_{\xi\xi}\nabla_\xi L_\mu.$$

$$H^{\text{eff}}_{pp}\,\delta p = -\nabla^{\text{eff}}_p L_\mu,$$
$$H_{\xi\xi}\delta\xi = -\nabla_\xi L_\mu - H_{\xi p}\delta p.$$

Extremize p in a direction $\vec{v}$

- Stationary conditions

$$L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0 \text{ and } \lambda v = \nabla_p L_\mu \text{ and } \lambda > 0.$$

- Newton update equations

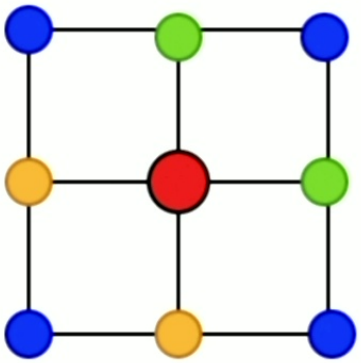
$$\frac{1}{2}\left(\delta p H_{pp}\delta p + \delta p H_{p\xi}\delta\xi + \delta\xi H_{\xi p}\delta p + \delta\xi H_{\xi\xi}\delta\xi\right) +$$
$$(\nabla_p L_\mu)^T \delta p + (\nabla_\xi L_\mu)^T \delta\xi = -L_\mu,$$
$$H_{\xi p}\delta p + H_{\xi\xi}\delta\xi = -\nabla_\xi L_\mu,$$
$$H_{pp}\delta p + H_{p\xi}\delta\xi = -\nabla_p L_\mu + \lambda v.$$

- Solve for Newton step

$$\frac{1}{2}\delta p^T H^{\text{eff}}_{pp}\delta p + (\nabla^{\text{eff}}_p L_\mu)^T \delta p = -L_\mu + (\nabla_\xi L_\mu)^T H^{-1}_{\xi\xi}(\nabla_\xi L_\mu)$$
$$H^{\text{eff}}_{pp}\,\delta p = -\nabla^{\text{eff}}_p L_\mu + \lambda v.$$

Compute Gradients and Hessians

We compute $(\nabla_p L_\mu)$, H_{pp} and $H_{p\xi}$ by taking the finite differences of $L(x, y, X, Y, b(p), c(p), B(p))$, requiring computing SDPs of a constellation of $n(n-1)/2, n = \dim(p)$ points.



Path to files is required as an input to the program.

Problematic Scenarios

- Non-convexity of $L(\xi, p)$ in p space.
  - high dimensions of ξ can potentially make $L(\xi, p)$ even more ill-conditioned.
- Interior point method, a changing objective function $L_\mu(\xi, p)$.
  - the quadratic mode constructed might not be reliable
- The solver stalls when the step lengths α's become too small for solutions to converge.

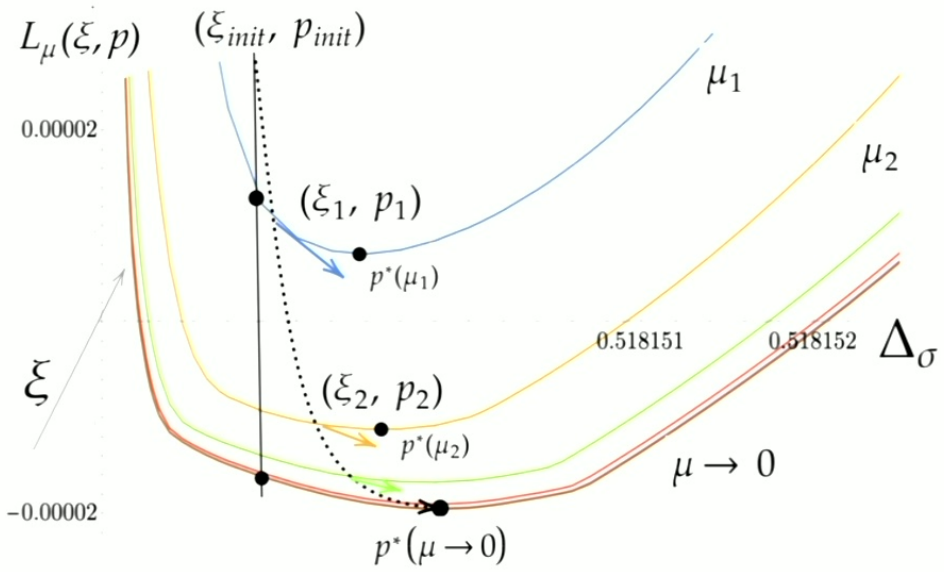
$$X + \alpha_P dX \geq 0, \quad Y + \alpha_P dY \geq 0.$$

  - α is an indicator of how close the solver is to the boundary of the positive region.
  - The barrier function $\mu \log\det X$ smooths the problem and pushes the solver to the interior but when this fails, the solver stalls.

Skydiving Algorithm: A Dynamical SDP Solver

Warm-up

Problematic Scenarios



$$L_\mu(\xi, p) \to (d\xi, dp),\ \mu' < \mu$$
$$\to \ldots \to L_{\mu=0}(\xi^*(\mu = 0), p^*(\mu = 0))$$

Problematic Scenarios

- Non-convexity of $L(\xi, p)$ in p space.
  - high dimensions of ξ can potentially make $L(\xi, p)$ even more ill-conditioned.
- Interior point method, a changing objective function $L_\mu(\xi, p)$.
  - the quadratic mode constructed might not be reliable
- The solver stalls when the step lengths α's become too small for solutions to converge.

$$X + \alpha_P dX \geq 0, \quad Y + \alpha_P dY \geq 0.$$

  - α is an indicator of how close the solver is to the boundary of the positive region.
  - The barrier function $\mu \log\det X$ smooths the problem and pushes the solver to the interior but when this fails, the solver stalls.

Problematic Scenarios

- Non-convexity of $L(\xi, p)$ in p space.
  - high dimensions of ξ can potentially make $L(\xi, p)$ even more ill-conditioned.
- Interior point method, a changing objective function $L_\mu(\xi, p)$.
  - the quadratic mode constructed might not be reliable
- The solver stalls when the step lengths α's become too small for solutions to converge.

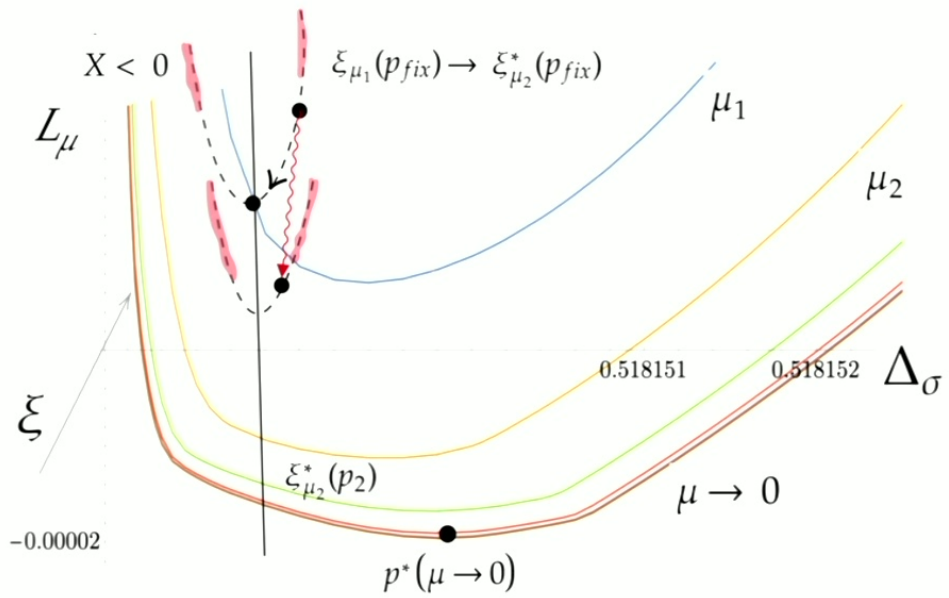
$$X + \alpha_P dX \geq 0, \quad Y + \alpha_P dY \geq 0.$$

  - α is an indicator of how close the solver is to the boundary of the positive region.
  - The barrier function $\mu \log\det X$ smooths the problem and pushes the solver to the interior but when this fails, the solver stalls.

Skydiving Algorithm: A Dynamical SDP Solver

Warm-up

Problematic Scenarios

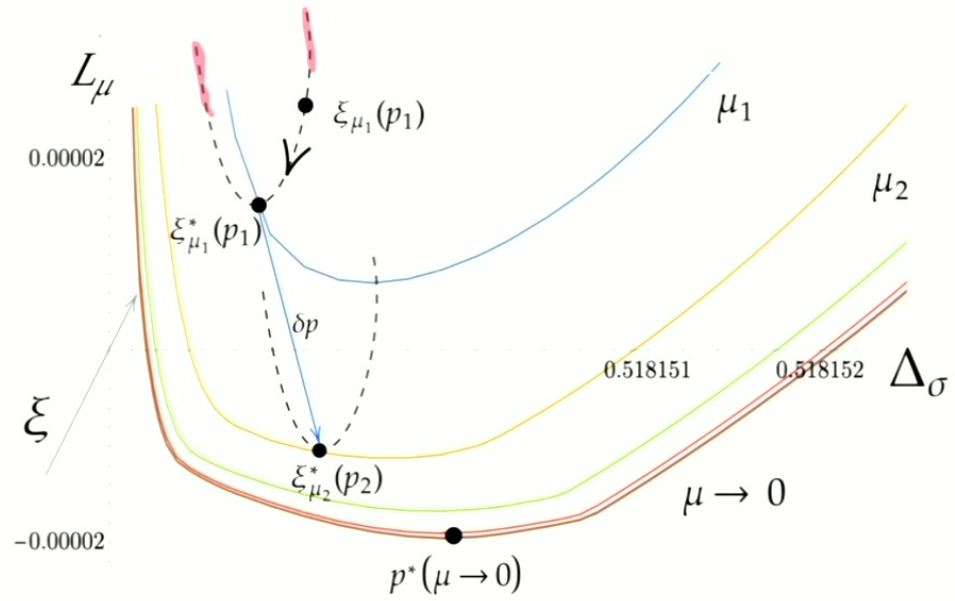


Fix p, decrease μ. Gradual decrease, stay positive.

Skydiving Algorithm: A Dynamical SDP Solver

Warm-up

Problematic Scenarios



$$L_\mu(\xi, p) \to (d\xi, dp),\ \mu' < \mu$$
$$\to \ldots \to L_{\mu=0}(\xi^*(\mu = 0), p^*(\mu = 0))$$

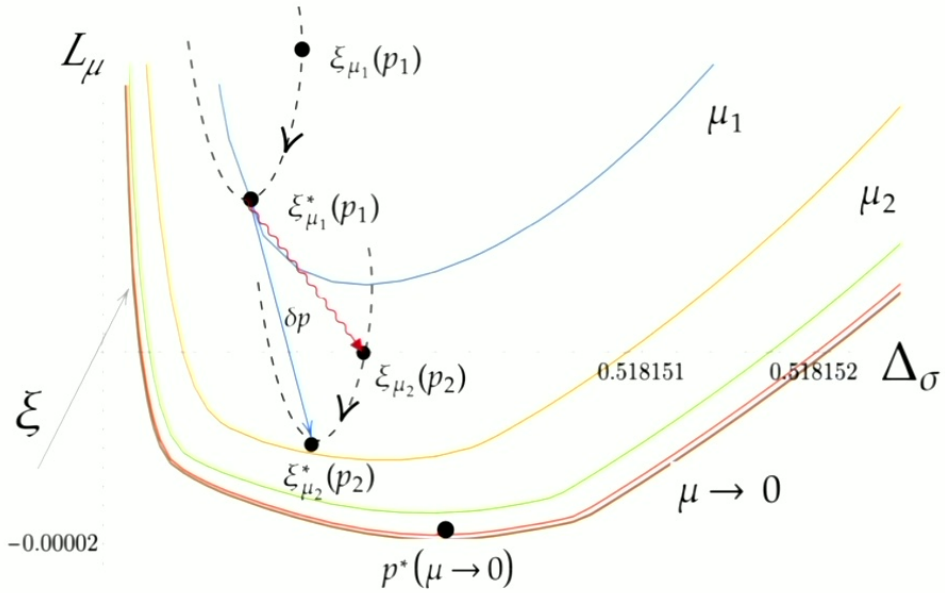
Solutions

- Centering: Stay within the interior of the positive region to avoid stalling.
- Scanning Algorithm: choose a trustworthy approximation to $L_\mu(\xi, p)$ to avoid stalling.
- A modified BFGS algorithm: avoid computing the expensive and possibly ill-conditioned H_{pp}.

Change p, decrease μ. Even more likely to hit negative regions.

Skydiving Algorithm: A Dynamical SDP Solver

Algorithm



We prioritize satisfying the stationary condition of ξ.
That is, at some (p, μ), we try to bring $\xi_\mu(p) \to \xi^*_\mu(p)$ where

$$\nabla_\xi L_\mu(\xi^*_\mu(p), p) = 0.$$

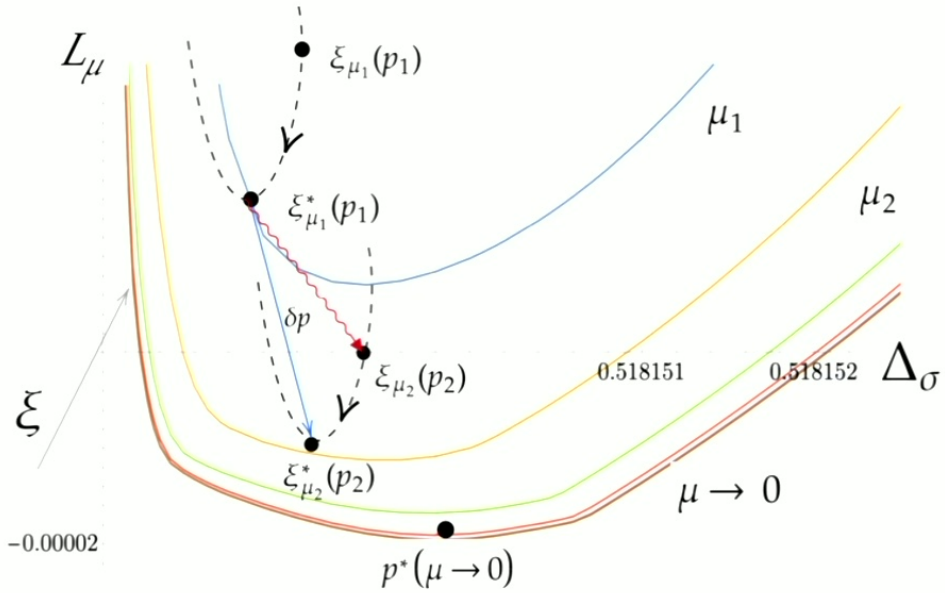
central section $\xi_\mu^*(p)$: $\nabla_\xi L_\mu(\xi_\mu^*(p), p) = 0$.

Avoid stalling
$\simeq \xi_\mu(p)$ stays in the "center" of the positive region as close as possible.
$\simeq \xi_\mu(p)$ is sufficiently close to $\xi_\mu^*(p)$.

At fixed p and μ, the error $\mathrm{Tr}\,(R)$ where $R := XY - \mu I$ is a good measure of how close $\xi_\mu(p)$ is to the optimal solution $\xi_\mu^*(p)$.

Skydiving Algorithm: A Dynamical SDP Solver

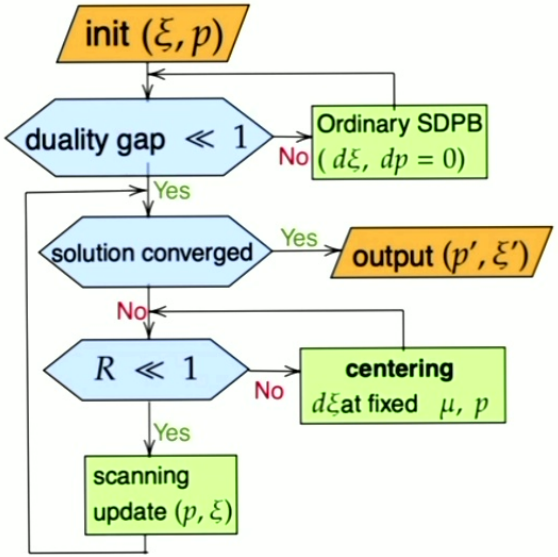
Algorithm



We prioritize satisfying the stationary condition of ξ.
That is, at some (p, μ), we try to bring $\xi_\mu(p) \to \xi^*_\mu(p)$ where

$$\nabla_\xi L_\mu(\xi^*_\mu(p), p) = 0.$$

Centering



- When

$$\mathrm{Tr}\,(R) \geq R_{\texttt{threshold}}$$

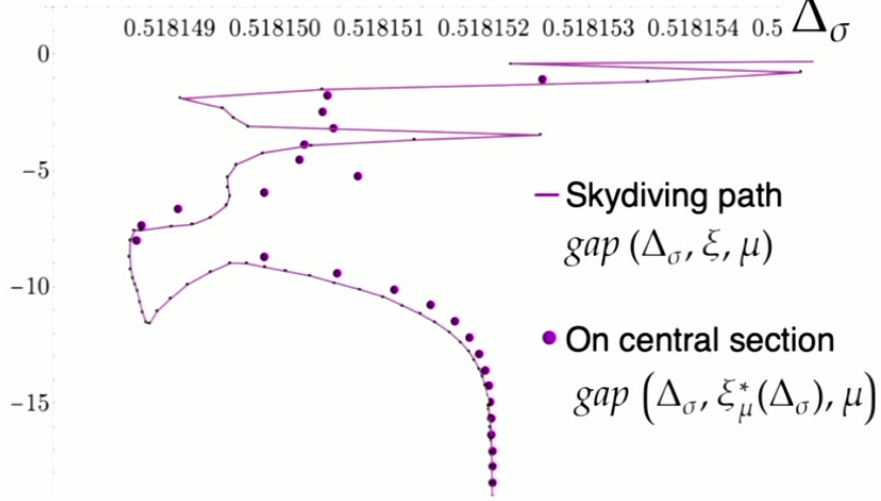
keep solving $\nabla_\xi L_\mu(\xi, p) = 0$ at fixed μ and p.

- Expect $O(1)$ number of centering steps. Significantly improve stalling problems.

After centering, we can focus on the curves $L_\mu(\xi^*(p), p)$ as functions of p.

Centering Example

log (*duality gap*), $gap \sim \mu$, $\mu_{next} = \beta\mu_{curr}$



$$\xi_\mu(p) \to \xi^*_\mu(p)$$

After centering, we can focus on the curves $L_\mu(\xi^*(p), p)$ as functions of p.

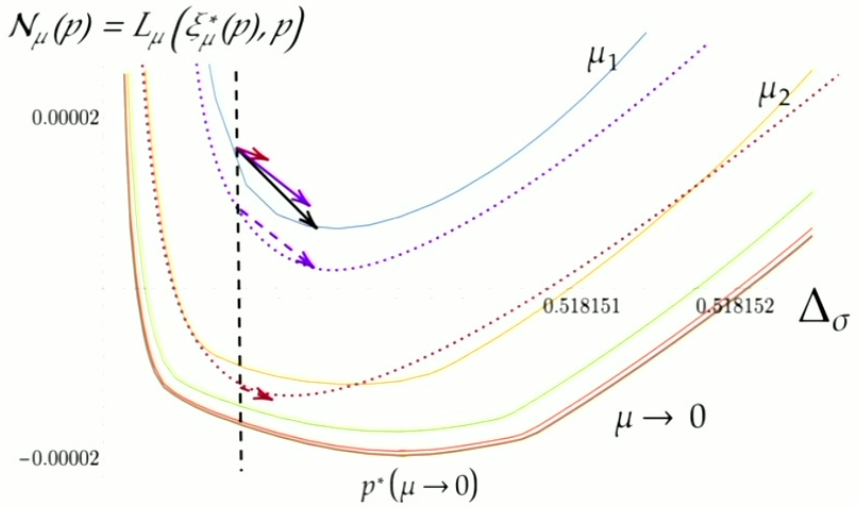
Scanning

At each iteration, we decide a new value of μ and compute the corresponding $(d\xi, dp)$, based on the quadratic model we constructed from $\nabla_p L_\mu$, $\nabla_\xi L_\mu$, H_{pp}, $H_{\xi p}$, and $H_{\xi\xi}$.

But how trustworthy is this quadratic model?

Scanning

- We want to decrease μ as quickly as possible for efficiency.
- But if the target μ is too small, the model is untrustworthy. It is possible that we take a "wrong" step and the solver might stall.

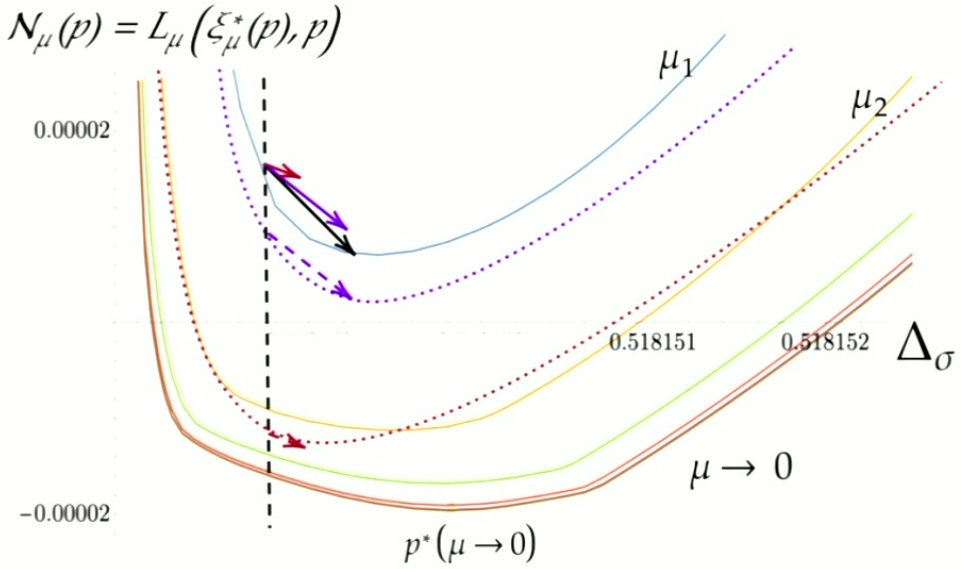


Skydiving Algorithm: A Dynamical SDP Solver

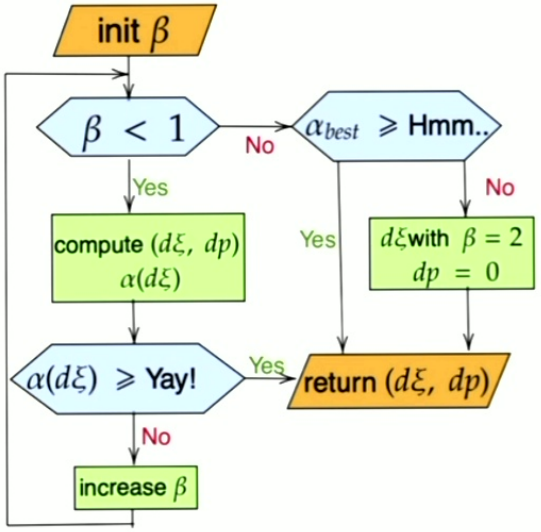
Algorithm

Scanning

- Use step lengths α_P, α_D as a criterion.
- α big enough $\simeq$ solver is not stalling $\simeq$ the model is trustworthy.



Scanning Algorithm



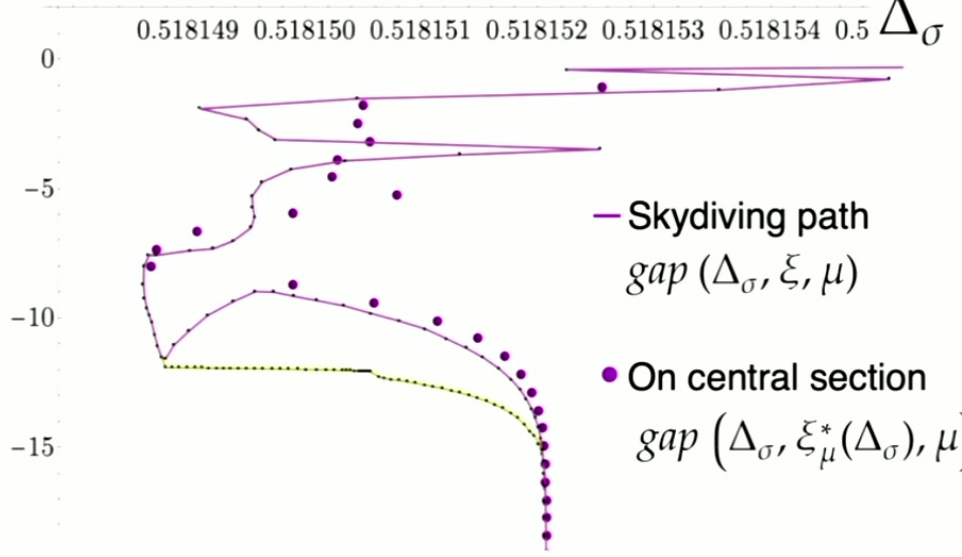
Yay! $\geq$ Hmm..

$\alpha_{best} \equiv \max_{\beta} \{\alpha(\beta)\}$

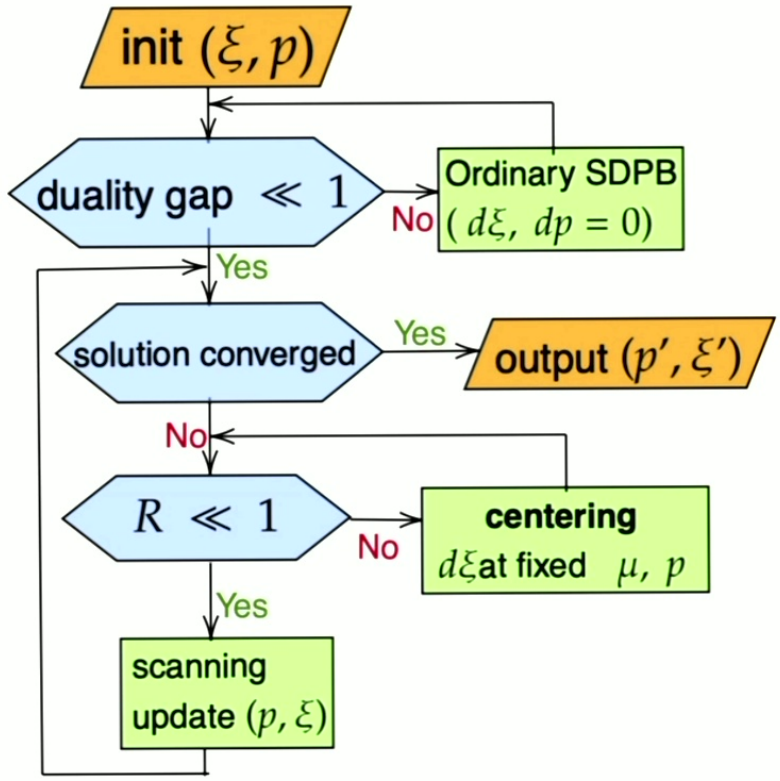
- compute $(\delta\xi, dp)$ using $\mu_{\texttt{next}} = (\beta_{\min} + i\Delta\beta)\mu_{\texttt{curr}}$.
- If $\max(\alpha_P, \alpha_D)$ is bigger than a user-chosen threshold, take this step. Otherwise, continue.
- When no valid step is found for all $\mu_{\texttt{next}} \leq \mu_{\texttt{curr}}$, increase $\mu_{\texttt{next}}$ further and give up updating p for this step.

Scanning Example

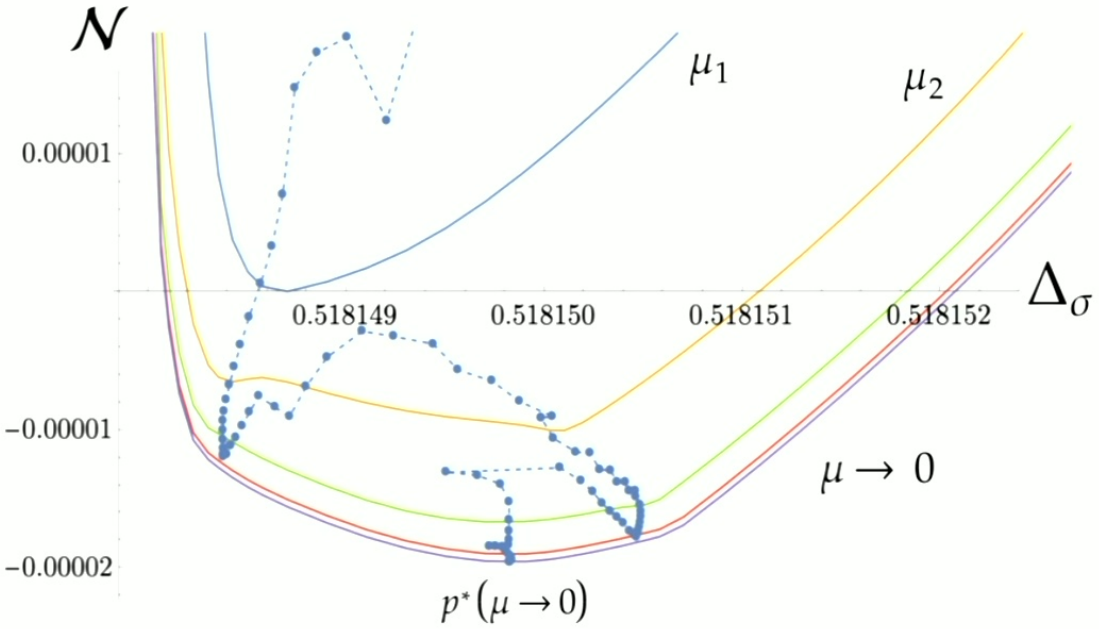
log (*duality gap*), $gap \sim \mu$, $\mu_{next} = \beta \mu_{curr}$



Skydiving Flow Chart



Full Skydiving Path



Revisiting BFGS

Motivations to use a quasi-Newton method to approximate the hessian matrix.

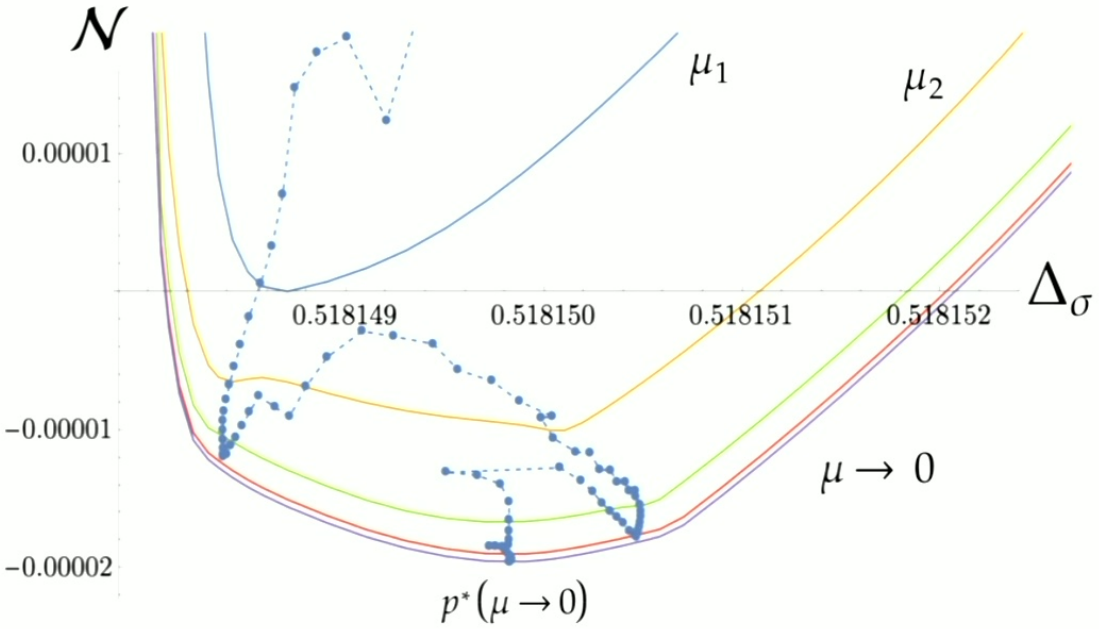
- Computing H_{pp} is expensive, need to produce $O(n^2)$ SDP files. A quasi-Newton method reduces it to $O(n)$.
- The exact Hessian matrix can be ill-conditioned, e.g., it is not positive definite. Then the exact method might not converge (fast).

Standard BFGS update,

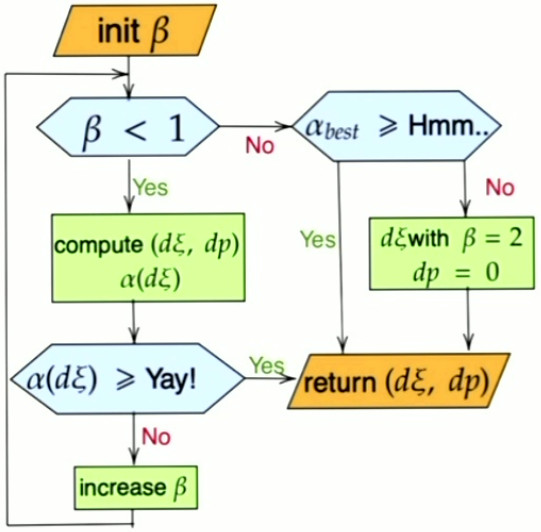
$$s_k = x_k - x_{k-1}, \quad y_k = \nabla_x f(x_k) - \nabla_x f(x_{k-1}),$$

$$H_k(x_k) = H_{k-1}(x_{k-1}) + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_{k-1} s_k s_k^T H_{k-1}^T}{s_k^T H_{k-1} s_k}.$$

Full Skydiving Path



Scanning Algorithm



Yay! $\geq$ Hmm..

$\alpha_{best} \equiv \max_{\beta} \{\alpha(\beta)\}$

- compute $(\delta\xi, dp)$ using $\mu_{\texttt{next}} = (\beta_{\min} + i\Delta\beta)\mu_{\texttt{curr}}$.
- If $\max(\alpha_P, \alpha_D)$ is bigger than a user-chosen threshold, take this step. Otherwise, continue.
- When no valid step is found for all $\mu_{\texttt{next}} \leq \mu_{\texttt{curr}}$, increase $\mu_{\texttt{next}}$ further and give up updating p for this step.

Revisiting BFGS

Motivations to use a quasi-Newton method to approximate the hessian matrix.

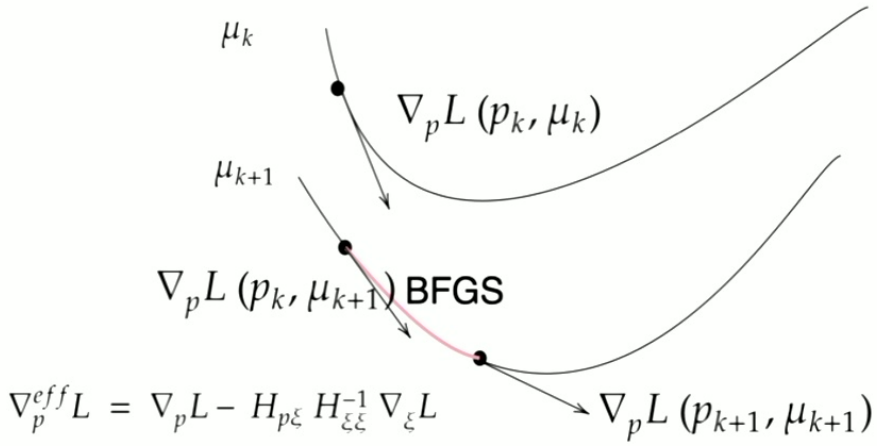
- Computing H_{pp} is expensive, need to produce $O(n^2)$ SDP files. A quasi-Newton method reduces it to $O(n)$.
- The exact Hessian matrix can be ill-conditioned, e.g., it is not positive definite. Then the exact method might not converge (fast).

Standard BFGS update,

$$s_k = x_k - x_{k-1}, \quad y_k = \nabla_x f(x_k) - \nabla_x f(x_{k-1}),$$

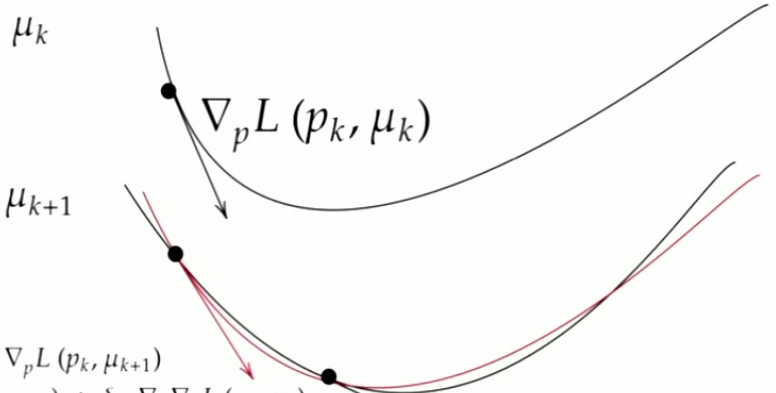
$$H_k(x_k) = H_{k-1}(x_{k-1}) + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_{k-1} s_k s_k^T H_{k-1}^T}{s_k^T H_{k-1} s_k}.$$

A moving objective function



- After centering $\nabla_\xi L_\mu = 0$, $\nabla_p^{\text{eff}} L \approx \nabla_p L$.
- Standard BFGS,
  $\nabla_p L(p_{k+1}, \mu_{k+1}) - \nabla_p L(p_k, \mu_{k+1}) \Rightarrow H_{pp}(p_{k+1}, \mu_{k+1})$.
- We have $\nabla_p L(p_k, \mu_k)$.
- Need to extrapolate $\nabla_p L(p_k, \mu_k) \to \nabla_p L(p_k, \mu_{k+1})$.

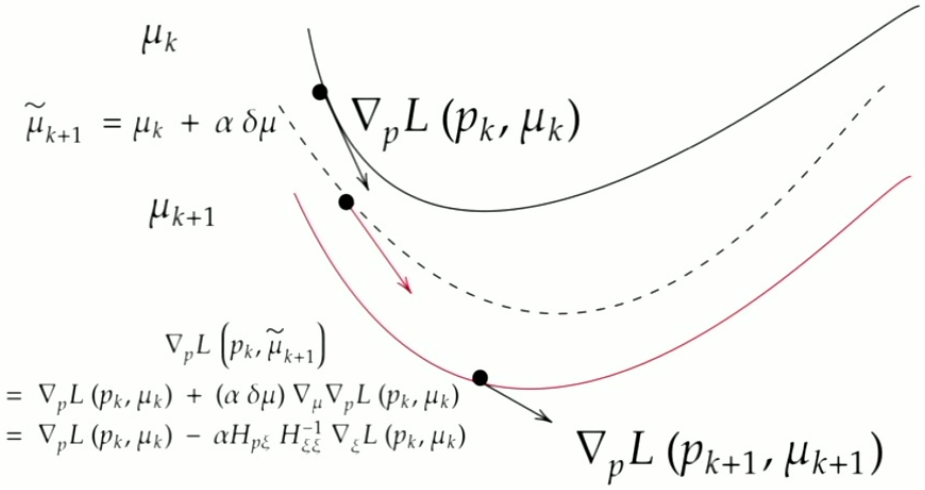
Modified BFGS algorithm



$$
\begin{aligned}
&\nabla_p L\,(p_k, \mu_{k+1}) \\
&= \nabla_p L\,(p_k, \mu_k) \;+\; \delta\mu\, \nabla_\mu \nabla_p L\,(p_k, \mu_k) \\
&= \nabla_p L\,(p_k, \mu_k) \;+\; \delta\mu\, \frac{\partial \xi}{\partial \mu} \partial_\xi \nabla_p L\,(p_k, \mu_k) \\
&= \nabla_p L\,(p_k, \mu_k) \;+\; H_{p\xi}\, \delta_\mu \xi \\
&= \nabla_p L\,(p_k, \mu_k) \;-\; H_{p\xi}\, H_{\xi\xi}^{-1}\, \nabla_\xi L\,(p_k, \mu_k) \\
&= \nabla_p^{eff} L\,(p_k, \mu_k)
\end{aligned}
$$

- $\nabla_p L(p_k, \mu_{k+1}) \approx \nabla_p^{\text{eff}} L(p_k, \mu_k)$, note this is before centering.
- $\nabla_p L(p_{k+1}, \mu_{k+1}) - \nabla_p^{\text{eff}} L(p_k, \mu_{k+1}) \Rightarrow H_{pp}^{\text{eff}}(p_{k+1}, \mu_{k+1})$.

Modified BFGS algorithm



- Consider the rescaling factor α, we reduce μ by $\alpha \cdot (\mu_{k+1} - \mu_k)$ rather than $(\mu_{k+1} - \mu_k)$.

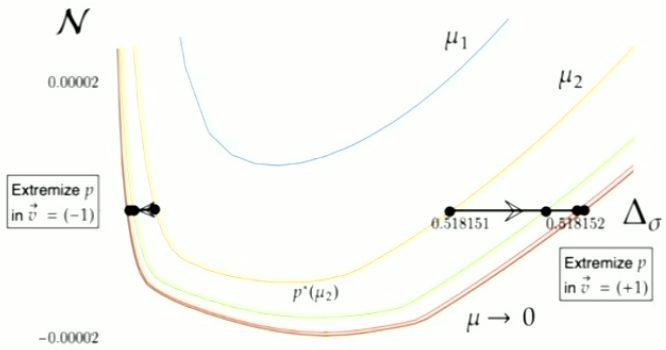
$$\nabla_p^{\text{eff}} L(p_k, \mu_{k+1}) = \nabla_p^L(p_k, \mu_k) - \alpha H_{p\xi} H_{\xi\xi}^{-1} \nabla_\xi L(p_k, \xi_k, \mu_{k+1})$$

Extremize p: Shifted Lagrangian

Stationary conditions when extremizing p in some direction $\vec{v}$,

$$L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0 \text{ and } \lambda v = \nabla_p L_\mu \text{ and } \lambda > 0$$

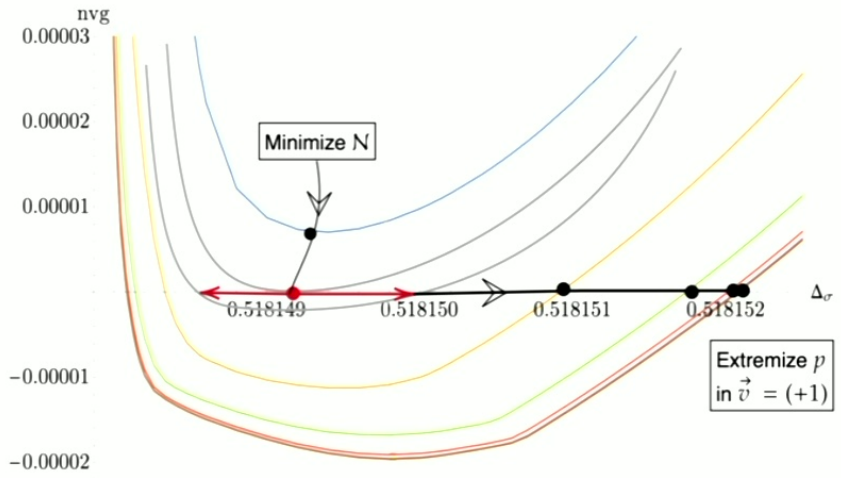
No solution when μ is large.



Algorithm design:

- Starts by minimizing L_μ, $\nabla_p L_\mu = 0$.
- Switch to extremizing p only when at current (ξ, p), $L_\mu(\xi, p) \leq 0$.

Extremize p: Shifted Lagrangian



But this creates problems too. For functions that is flat near the minimum, a big jump can be incurred by switching the search target. Such discontinuity can lead to stalling.

Extremize p: A Constant Shift

Remember μ smooths the problem, so if there has to be a jump, jump as early as possible.

$$L_\mu(x, y, X, Y) = c^T x + b^T y - x^T B y + \mathrm{Tr}\left((X - x^T A_*) Y\right) - \mu \log\det X$$

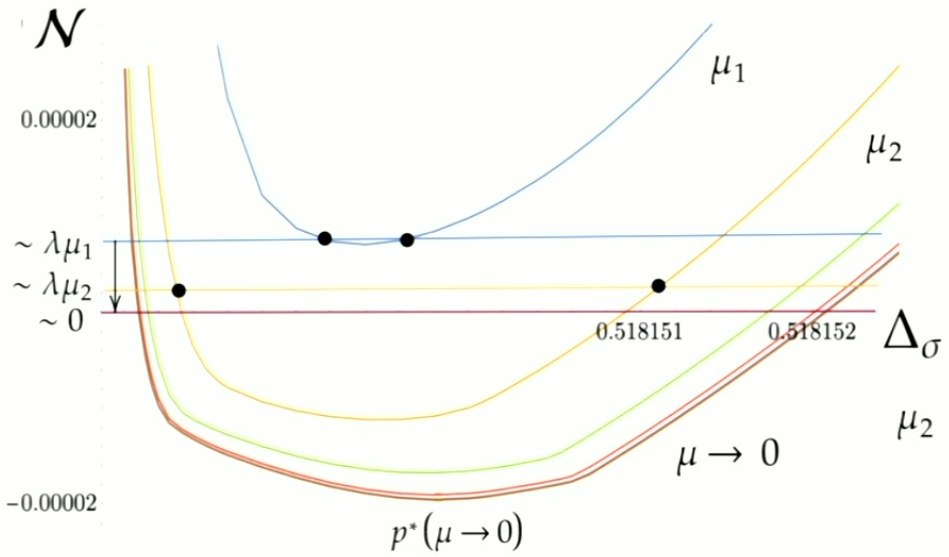
The solution is to shift $L_\mu(\xi^*(p), p)$ by a number proportional to μ

$$L_\mu(x, y, X, Y) = \lambda \mu \dim(X)$$

Extremize p: A Constant Shift

$$L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0 \text{ and } \lambda v = \nabla_p L_\mu \text{ and } \lambda > 0$$

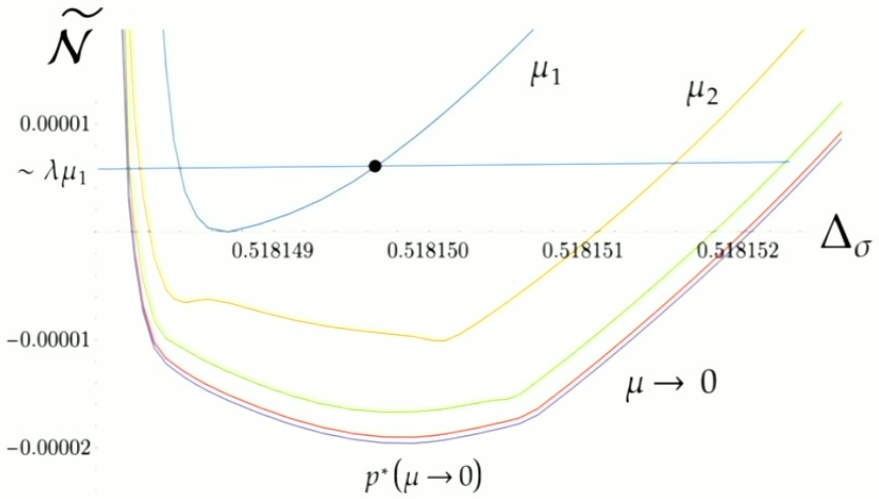
$$L_\mu = 0 \quad \Rightarrow \quad L_\mu(x, y, X, Y) = \lambda \mu \dim(X)$$



Extremize p: An X-dependent Shift

$$L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0 \text{ and } \lambda v = \nabla_p L_\mu \text{ and } \lambda > 0$$

$$L_\mu = 0 \quad \Rightarrow \quad L_\mu(x, y, X, Y) = \lambda\mu \dim(X) - \mu \log\det X$$



Shifted Path

$$L_\mu = 0 \text{ and } \nabla_\xi L_\mu = 0 \text{ and } \lambda v = \nabla_p L_\mu \text{ and } \lambda > 0$$

$$L_\mu = 0 \quad \Rightarrow \quad L_\mu(x, y, X, Y) = \lambda\mu \dim(X) - \mu \log\det X$$

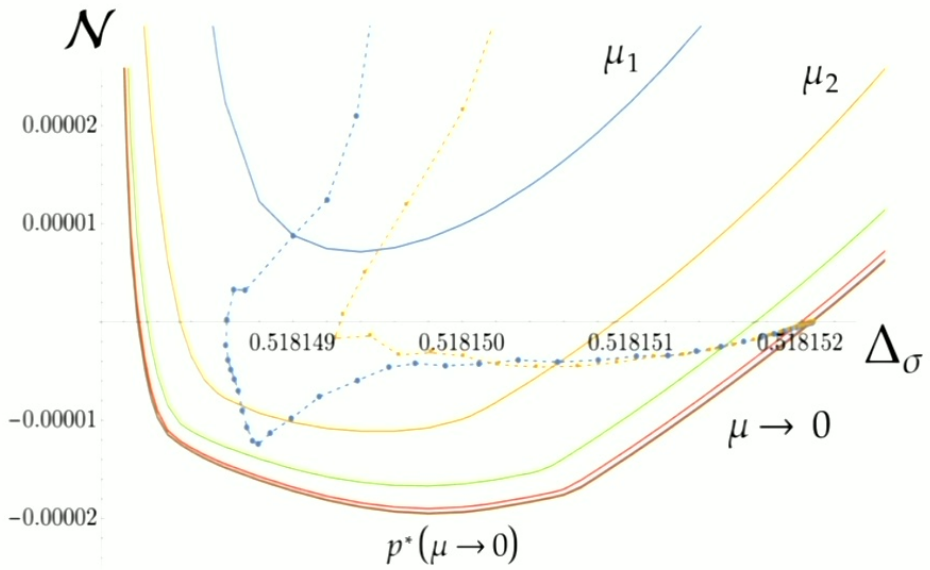
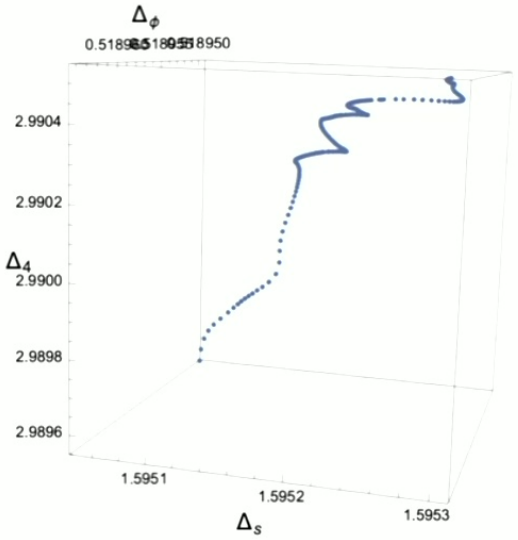


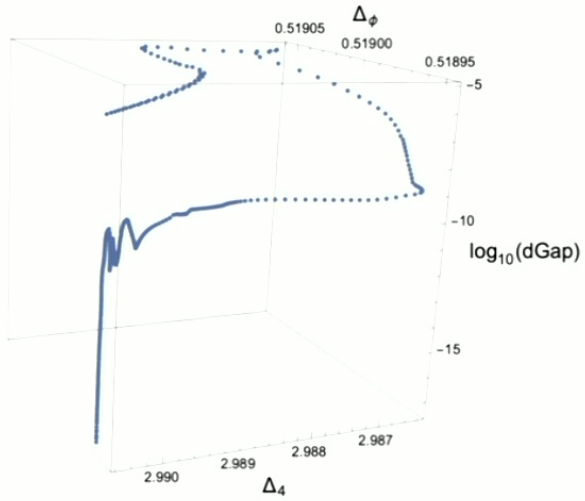
Table of Contents

Results and Performance

step	iter	dualityGap	$(\Delta_\sigma, \Delta_\epsilon, \theta)$
1	138	0.08717930312	v[0.518977, 1.40040, 0.955188]
2	4	0.08973379600	v[0.518955, 1.40078, 0.954438]
3	4	0.09245803804	v[0.518934, 1.40114, 0.953745]
4	4	0.09536990641	v[0.518914, 1.40147, 0.953112]
5	4	0.09532004688	v[0.518895, 1.40179, 0.952544]
6	4	0.09489738642	v[0.518878, 1.40208, 0.952036]
7	4	0.09449663964	v[0.518861, 1.40234, 0.951581]
8	4	0.09411554917	v[0.518846, 1.40259, 0.951174]
9	4	0.09374992985	v[0.518831, 1.40282, 0.950805]
10	4	0.09339396813	v[0.518818, 1.40303, 0.950467]
11	4	0.09304120125	v[0.518804, 1.40324, 0.950155]
12	4	0.09268513482	v[0.518791, 1.40344, 0.949864]
13	4	0.09231924600	v[0.518778, 1.40364, 0.949590]
14	4	0.09193661096	v[0.518765, 1.40384, 0.949331]
15	4	0.09153302137	v[0.518753, 1.40403, 0.949092]
16	4	0.09112837036	v[0.518741, 1.40421, 0.948870]
17	4	0.09071110758	v[0.518728, 1.40440, 0.948659]
18	4	0.09026895055	v[0.518716, 1.40458, 0.948462]
19	4	0.08980798307	v[0.518702, 1.40479, 0.948232]

Results and Performance (O(3) tiptop

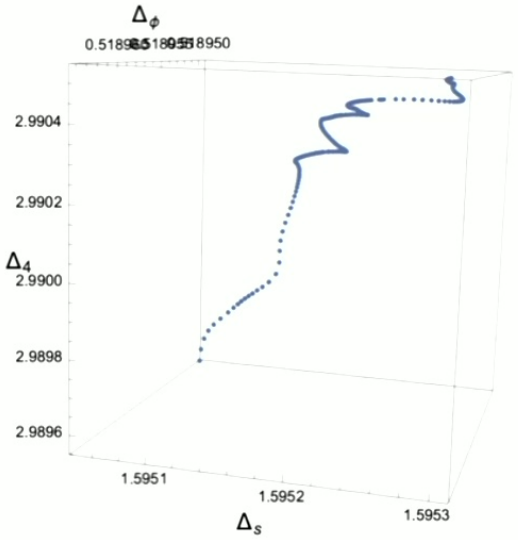


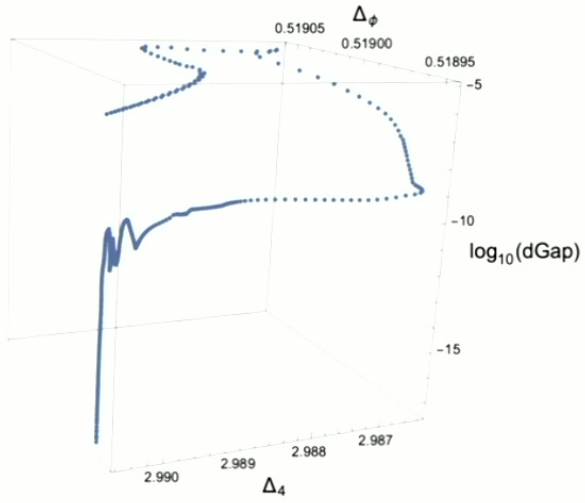


Results and Performance

Goals	Λ	p Dim	p
Ising (σ, ϵ) Minimize $\mathcal{N}$	11	3	$(\Delta_\sigma, \Delta_\epsilon, \theta)$
Ising (σ, ϵ) Maxmize Δ_σ	11	3	$\Delta_\sigma = 0.51975519225064..$
O(3) (ϕ, s, t) tiptop	19	8	$\Delta_{t_4} = 2.9998459435(2)$
O(3) (ϕ, s, t) tiptop	35	8	$\Delta_{t_4} = 2.990548(1)$
Previous O(3) (ϕ, s, t) tiptop	35	8	$\Delta_{t_4} < 2.99056$

Results and Performance (O(3) tiptop



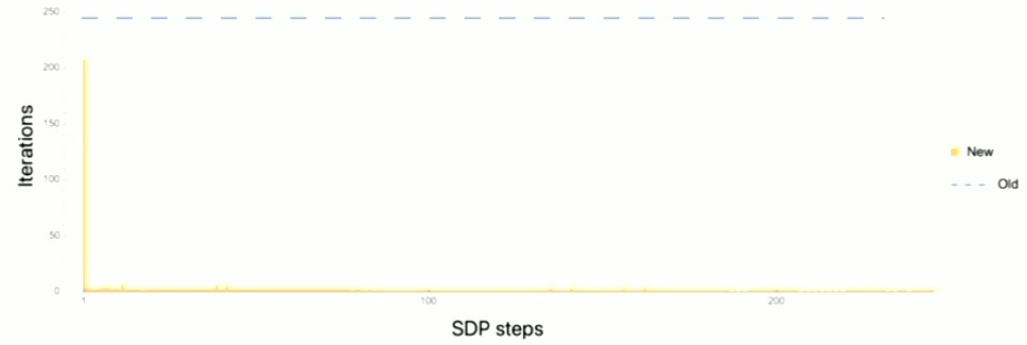


Results and Performance

Goals	Λ	Gap	Preset Prec	CPU h's
Ising (σ, ϵ) Minimize $\mathcal{N}$	11	10^{-30}	768	48
Ising (σ, ϵ) Maxmize Δ_σ	11	10^{-30}	768	48
O(3) (ϕ, s, t) tiptop	19	10^{-25}	768	–
O(3) (ϕ, s, t) tiptop	35	10^{-18}	448	84532
Previous O(3) (ϕ, s, t) tiptop	35	–	960	≥ 1 M

Total CPU hours= 36051.7 (SDP solver) + 48480 (generate SDPs), where 11024.7 out of 36051.7 solver hours come from loading ξ.

Results and Performance



Limitatinos

- Situations when `Centering` fails to reduce R.