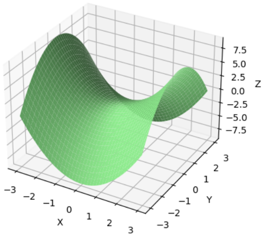
Disciplined Saddle Programming

Philipp Schiele, Eric Luxenberg, Stephen Boyd

July 12–2023

SciPy Conference, Austin, TX

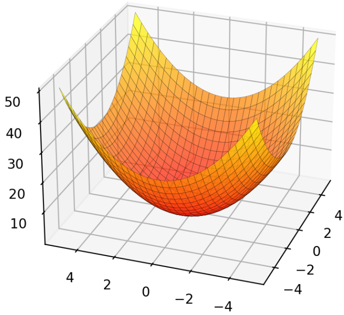


Disciplined Saddle Programming (DSP)

- Domain specific language for saddle programming.
- Implemented as an extension to CVXPY.
- Method based on recent work by Juditsky and Nemirovski [JN22].
- Natural use case is robust optimization.

CVXPY

- CVXPY is a Python-embedded modeling language for convex optimization.



Convex optimization problem

Formally, a *convex optimization problem* is can be written as

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- variable $x \in \mathbf{R}^n$
- equality constraints are linear
- $f_0, \ldots, f_m$ are **convex**: for $\theta \in [0, 1]$,

$$f_i(\theta x + (1 - \theta) y) \le \theta f_i(x) + (1 - \theta) f_i(y)$$

i.e., f_i have nonnegative (upward) curvature

Solving convex problems with CVXPY

- CVXPY allows to solve convex optimization problems in a natural way.

```python
import cvxpy as cp

x = cp.Variable(2)
objective = cp.sum_squares(x)
constraints = [-4 <= x, x <= 4]
problem = cp.Problem(cp.Minimize(objective), constraints)
opt_val = problem.solve()  # 0.0
solution = x.value  # array([0., 0.])
```

Linear programming

- Let us consider the simple linear program

$$
\begin{array}{lrcrl}
\text{maximize} & 2x_1 & + & 3x_2 & \\
\text{subject to} & x_1 & & & \leq 5 \\
 & & & x_2 & \leq 4 \\
 & x_1 & + & x_2 & \leq 7
\end{array}
$$

Finding an upper bound

- Can we combine the constraints to find an upper bound?

$$
\begin{array}{lrcrll}
\text{maximize} & 2x_1 & + & 3x_2 & & \\
\text{subject to} & x_1 & & & \leq 5 & |(*1) \\
& & & x_2 & \leq 4 & |(*2) \\
& x_1 & + & x_2 & \leq 7 & |(*1) \\
\hline
& 2x_1 & + & 3x_2 & \leq 20 &
\end{array}
$$

Finding the smallest upper bound

- Can we combine the constraints to find an upper bound?

$$\begin{array}{lrcrll} \text{maximize} & 2x_1 & + & 3x_2 & & \\ \text{subject to} & x_1 & & & \leq 5 & |(*y_1) \\ & & & x_2 & \leq 4 & |(*y_2) \\ & x_1 & + & x_2 & \leq 7 & |(*y_3) \end{array}$$

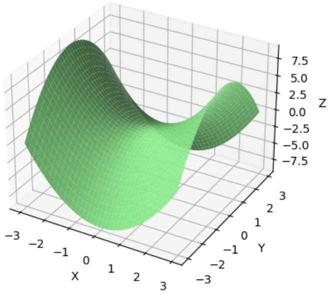
This means, we can write the problem as

$$\begin{array}{lrcrcrl} \text{minimize} & 5y_1 & + & 4y_2 & + & 7y_3 & \\ \text{subject to} & y_1 & + & & + & y_3 & \geq 2 \\ & & + & y_2 & + & y_3 & \geq 3 \\ & & & & & y_1, y_2, y_3 & \geq 0 \end{array}$$

- This is again a linear program.
- It is the so-called *dual* of the original problem.

Saddle function

- Convex optimization deals with functions that have a joint curvature in all their arguments.
- A *(convex-concave) saddle function* $f : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}$ is convex in $f(\cdot, y)$ for any fixed $y \in \mathcal{Y}$, concave in $f(x, \cdot)$ for any fixed $x \in \mathcal{X}$.



A saddle point problem

- A *saddle point problem* is to find a *saddle point* of a saddle function.
- A saddle point $(x^\star, y^\star) \in \mathcal{X} \times \mathcal{Y}$ is any point that satisfies

$$f(x^\star, y) \leq f(x^\star, y^\star) \leq f(x, y^\star) \text{ for all } x \in \mathcal{X},\ \ y \in \mathcal{Y}.$$

- In other words, $x^\star$ minimizes $f(x, y^\star)$ over $x \in \mathcal{X}$, and $y^\star$ maximizes $f(x^\star, y)$ over $y \in \mathcal{Y}$.

A simple example

- A *matrix game* is a game where two players choose strategies $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^n$, respectively.
- For a given payoff matrix C, the resulting payment is $f(x, y) = x^T C y$.
- The player choosing x wants to minimize the payment, and the player choosing y wants to maximize the payment.

A simple example ctd.

- Let us consider the following matrix game with variable $x \in \mathbf{R}^2$ and $y \in \mathbf{R}^2$.

	y_1	y_2
x_1	1	2
x_2	3	1

We restrict $x_1, x_2, y_1, y_2 \geq 0$ and $x_1 + x_2 = 1$, $y_1 + y_2 = 1$.
Recall that x tries to minimize $x^T Cy$, and y tries to maximize.

A simple example ctd.

- Matrix games can be solved as a convex optimization problem by dualizing the problem.
- This solution method goes back to Von Neumann and Morgenstern [MVN53].
- However, dualizing the problem requires working knowledge of duality, and is error prone.
- DSP allows formulating this problem explicitly as a saddle point problem.

The matrix game in DSP

```python
import dsp

x = cp.Variable(2, nonneg=True)
y = cp.Variable(2, nonneg=True)
C = np.array([[1, 2], [3, 1]])

f = dsp.inner(x, C @ y)
obj = dsp.MinimizeMaximize(f)

constraints = [cp.sum(x) == 1, cp.sum(y) == 1]
prob = dsp.SaddlePointProblem(obj, constraints)
prob.solve()

prob.value  # 1.6666666666666667
x.value  # array([0.66666667, 0.33333333])
y.value  # array([0.33333333, 0.66666667])
```

Conic standard form as an API

- Many convex optimization problems can be written in the following form:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \mathcal{K} \end{array}$$

- This allows for a separation of concerns between
  - Modeling languages
  - Solvers
  - Research about algorithms

Conic standard form as an API ctd.

- Use CVXPY as a tool to obtain conic representation of saddle functions.
- Apply automated dualization to obtain single minimization problem.
- Use CVXPY to solve the resulting problem.

$$
\begin{aligned}
\Phi(x) &= \sup_{y \in \mathcal{Y}} \phi(x,y) \\
&= \sup_{y \in \mathcal{Y}} \inf_{f,t,u} \left\{ f^T y + t \;\middle|\; Pf + tp + Qu + Rx \preceq_K s \right\} \\
&= \inf_{f,t,u} \left\{ \sup_{y \in \mathcal{Y}} \left(f^T y + t \right) \;\middle|\; Pf + tp + Qu + Rx \preceq_K s \right\} \\
&= \inf_{f,t,u} \left\{ \sup_{y \in \mathcal{Y}} \left(f^T y \right) + t \;\middle|\; Pf + tp + Qu + Rx \preceq_K s \right\} \\
&= \inf_{f,t,u} \left\{ \inf_{\lambda} \left\{ \lambda^T e \;\middle|\; \begin{array}{l} C^T\lambda = f,\ D^T\lambda = 0 \\ \lambda \succeq_{K^*} 0 \end{array} \right\} \;\middle|\; Pf + tp + Qu + Rx \preceq_K s \right\}
\end{aligned}
$$

Example: Rocket landing

- We showed how to solve a rocket landing problem using CVXPY on Monday.
- The objective was to minimize the fuel used to land a rocket.

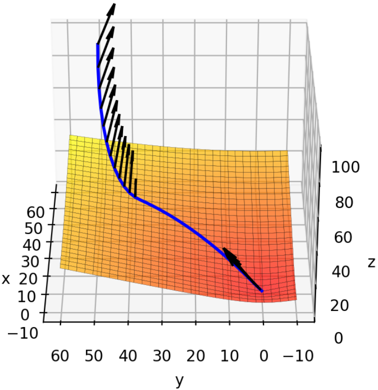
```python
V = cp.Variable((K + 1, 3))  # velocity
P = cp.Variable((K + 1, 3))  # position
F = cp.Variable((K, 3))  # thrust

constraints = [...]

fuel_consumption = gamma * cp.sum(cp.norm(F, axis=1))

problem = cp.Problem(cp.Minimize(fuel_consumption), constraints)
problem.solve()
```

Example: Rocket landing ctd.



This trajectory uses 150t of fuel.

Example: Robust rocket landing

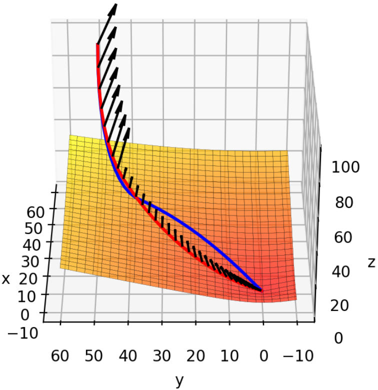
- We now assume γ is the *average* fuel consumption.
- In each period, $\hat{\gamma}_k$ can be within $\gamma \pm 30\%$.
- We want to find the best trajectory for the worst case $\hat{\gamma}$.

```
gamma_hat = cp.Variable(K)
constraints += [
        cp.sum(gamma_hat)/K == gamma,
        0.7 * gamma <= gamma_hat, gamma_hat <= gamma * 1.3
]

fuel_consumption_saddle = dsp.saddle_inner(cp.norm(F, axis=1), gamma_hat)

problem = dsp.SaddlePointProblem(
        dsp.MinimizeMaximize(fuel_consumption_saddle),
        constraints
)
problem.solve()
```

Example: Robust rocket landing ctd.



This trajectory uses 170t of fuel.

Saddle extremum functions

- A saddle extremum (SE) function is a partial supremum or infimum of a saddle function.
- The partial supremum is referred to as a *saddle max*

$$G(x) = \sup_{y \in \mathcal{Y}} f(x, y), \quad x \in \mathcal{X},$$

  with the partial infimum referred to as a *saddle min*.
- When only the objective is a SE, the problem we have a saddle point problem.
- A *saddle problem* more generally can include SE functions in its constraints.
- Since SE functions are convex (concave) expressions, they can be used in any CVXPY problem.

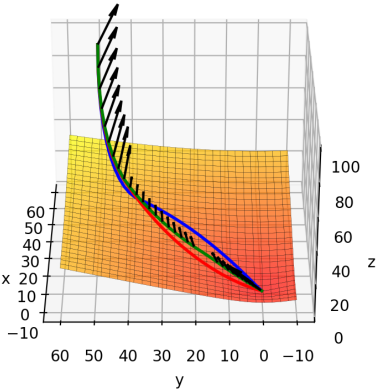
Example: Rocket landing with robust constraint

- Use the average fuel consumption γ as the objective.
- Want the worst case fuel consumption to be manageable.

```python
gamma_hat = dsp.LocalVariable(K, nonneg=True)
local_constraints += [
        cp.sum(gamma_hat)/K == gamma,
        0.7 * gamma <= gamma_hat, gamma_hat <= gamma * 1.3
]
fuel_consumption_saddle = dsp.saddle_inner(cp.norm(F, axis=1), gamma_var)
fuel_consumption_wc = dsp.saddle_max(
        fuel_consumption_dsp,
        local_constraints
)

constraints += [fuel_consumption_wc <= 175]
fuel_consumption = gamma * cp.sum(cp.norm(F, axis=1))
problem = cp.Problem(cp.Minimize(fuel_consumption), constraints)
```

Example: Rocket landing with robust constraint ctd.



This trajectory uses 152t of fuel.

Model comparison

- Robust constraint gives us a tradeoff between average and worst case fuel consumption.

Model	Nominal objective	Worst case objective
Nominal	150.2t	195.3t
Worst case	152.8t	170.2t
Robust constraint	151.7t	175.0t

Applications

- DSP can be used in many applications, including game theory, control, machine learning, and finance.
- Examples include
  - *Matrix games*.
  - *Robust rocket control*.
  - *Robust Markowitz portfolio optimization*.
  - *Robust bond portfolio optimization*.
  - *Robust regression model fitting*.
  - ...
- Where else can DSP be used? Let us know!

Getting started

- DSP is available on GitHub cvxgrp/dsp.
- The paper is on arxiv arXiv:2301.13427.
- Try it out now: `pip install dsp-cvxpy`.

Resources

- *Convex Optimization* (book)
- *EE364a* (course slides, videos, code, homework, ...)
- software CVXPY, CVX, Convex.jl, CVXR
- *convex optimization short course*
- *The Art of Linear Programming* [on YouTube]

all available online

References I

A. Juditsky and A. Nemirovski.
On well-structured convex–concave saddle point problems and variational inequalities with monotone operators.
Optimization Methods and Software, 37(5):1567–1602, 2022.

O. Morgenstern and J. Von Neumann.
Theory of games and economic behavior.
Princeton University Press, 1953.

Backup slides

Composition rules

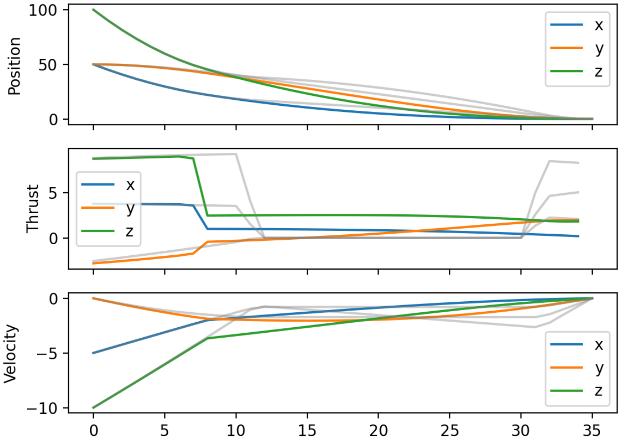
- Every DCP function is convex, but not every convex function is DCP.
- Likewise, every DSP function is a saddle function, but not every saddle function is DSP.
- To construct a DSP function, we start from DSP atoms, which includes all DCP atoms.
- Saddle functions can be be scaled and composed by addition.
- When adding two saddle functions, a variable may not appear as a convex variable in one expression and as a concave variable in the other expression.

Manual dualization in CVXPY

- Some atoms in CVXPY are implemented as manual dualizations.
- As an example, take the *sum of k largest entries* atom.
- This atom can be represented as a partial supremum of the saddle function $f(x, y) = x^T y$, with $\mathcal{Y} = \{y \mid 0 \leq y \leq 1,\ 1^T y = k\}$.
- DSP automates the dualization, such that sum of k largest entries can be written as

```
x = cp.Variable(n)
y = dsp.LocalVariable(n, nonneg=True)
f = dsp.inner(x, y)
constraints = [y <= 1, cp.sum(y) == k]
sum_k_largest = dsp.saddle_max(f, constraints)
```

Thrust, velocity, and position for robust rocket landing



Thrust, velocity, and position for robust constrained rocket landing