

Flexible Toolbox for Frank-Wolfe Algorithms & Application to Generalized Self-concordance

Mathieu Besançon, Alejandro Carderera, Sebastian Pokutta

07.07.2021

Zuse Institute Berlin - AIS²T

Frank-Wolfe algorithms

FrankWolfe.jl: a Julia implementation

Frank-Wolfe and generalized self-concordance

Frank-Wolfe algorithms

FrankWolfe.jl: a Julia implementation

Frank-Wolfe and generalized self-concordance

Considered problems for Frank-Wolfe

Problem class considered:

$$(P) : \min_x f(x) \\ \text{s.t. } x \in C$$

with:

- C : compact convex set,
- f : continuously differentiable on C .

Key requirement:

Optimizing a linear function over C much cheaper than (P) itself.

Linear Minimization Oracle (LMO):

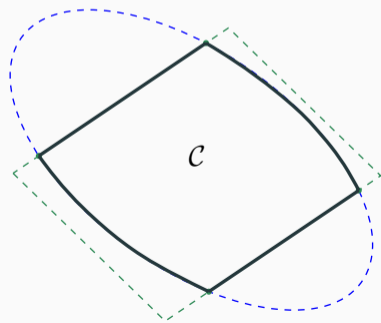
$$d \rightarrow v \in \arg \min_{y \in C} \langle y, d \rangle.$$

Frank-Wolfe structure and conic optimization

Frank-Wolfe:

$$(P) : \min_x f(x) \text{ s.t. } x \in C$$

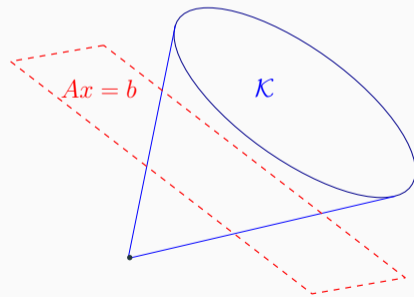
C compact convex (bounded).



Conic optimization:

$$(C) : \min_x \langle c, x \rangle \text{ s.t. } Ax = b, x \in \mathcal{K}$$

\mathcal{K} proper cone.

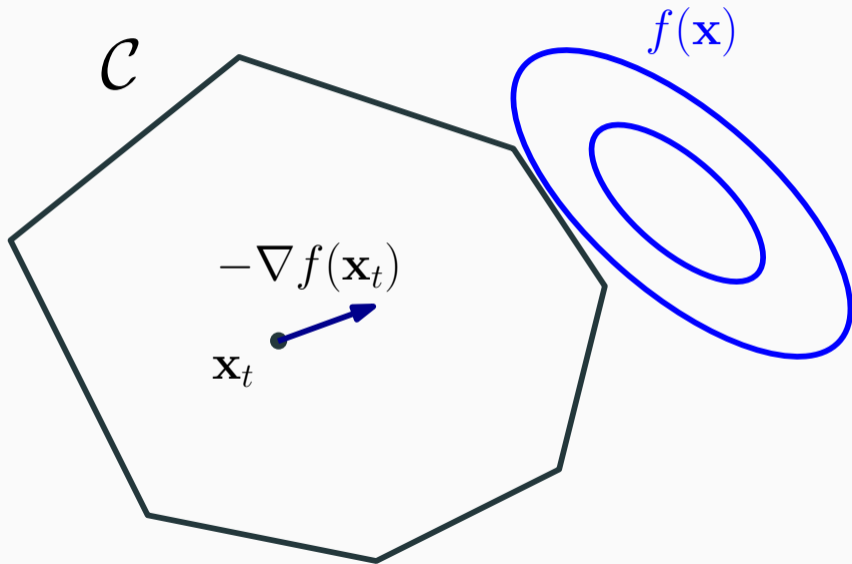


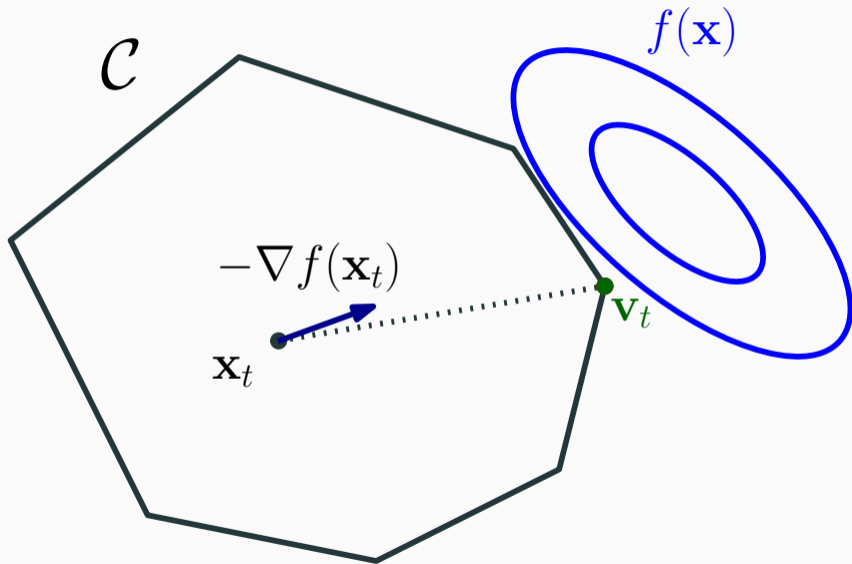
Algorithm 1.1 Frank-Wolfe algorithm

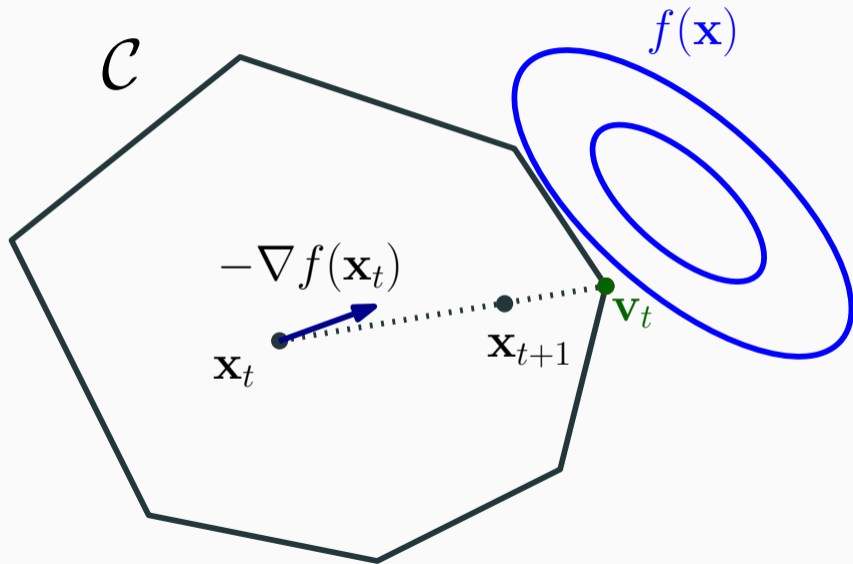
Require: Point $x_0 \in C$, function f .

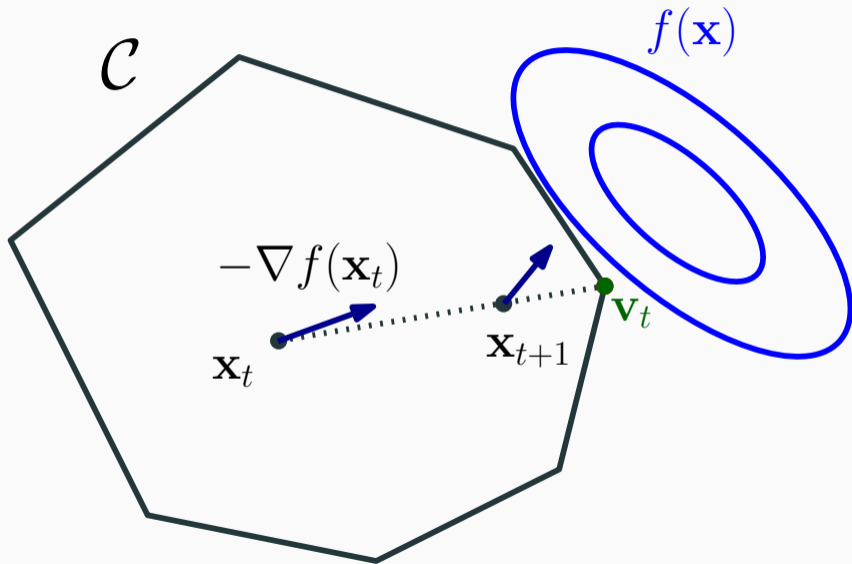
Ensure: Iterates $x_1, \dots \in C$.

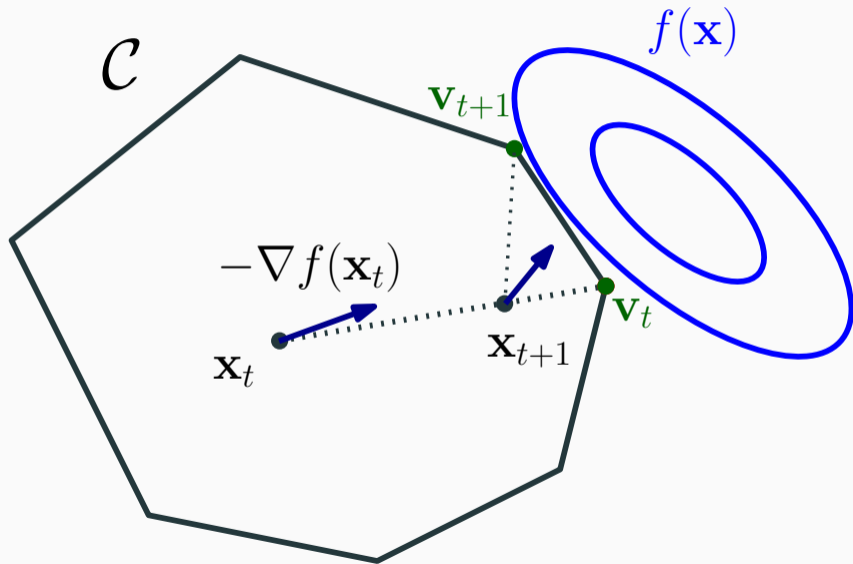
- 1: **for** $t = 0$ **to** \dots **do**
 - 2: $d_t \leftarrow \text{FirstOrderEstimate}(f, x_t)$
 - 3: $v_t \leftarrow \arg \min_{v \in C} \langle d_t, v \rangle$
 - 4: $\gamma_t \leftarrow \text{StepSizeStrategy}(x_t, t, d_t, v_t)$
 - 5: $x_{t+1} \leftarrow x_t + \gamma_t(v_t - x_t)$
 - 6: **end for**
-











How hard is optimizing a linear function?

Generic polytope \Rightarrow Linear Optimization method (simplex, interior points)

Many simple polytopes \Rightarrow closed-form solutions $\ell_{1,2,\infty}$ norm balls, simplex. . .

Requirement: black box computation of primal value only \rightarrow convex hull of

Mixed-Integer Problems.

Frank-Wolfe algorithms

FrankWolfe.jl: a Julia implementation

Frank-Wolfe and generalized self-concordance

FrankWolfe.jl: yet another implementation?

FW algorithms:

Deceptive simplicity but no de-facto implementation.

Consequence: every paper reinventing the wheel, potential bugs, performance variability, etc.

Goal: one central toolbox for:

1. Practitioners solving optimization problems fitting the form (P) ,
2. Researchers on Frank-Wolfe type algorithms developing new methods.

One-slide package summary:

Implemented in Julia:

- Compiled to native code, reaches C-like performance;
- Highly generic thanks to multiple dispatch;
- Generic numeric types: reduced (16, 32, 64 bits) and extended (128, GNU MP) precision, rationals;
- Memory-saving mode, in-place gradient computations;
- Switchable components - bring your own LMO / gradient / step size.

Main variants

Variant	Convergence		Sparsity	Numerical Stability	Active Set?	Lazy?
	Progress/iter.	Time/iter.				
FW	Low	Low	Low	High	No	Yes
Away FW	Medium	Medium-High	Medium	Medium-High	Yes	Yes
Blended CG	High	Medium-High	High	Medium	Yes	By design
Stoch. FW	Low	Low	Low	High	No	No

Bring your own component:

$$1: d_t \leftarrow \text{FirstOrderEstimate}(f, x_t)$$

$$2: v_t \leftarrow \arg \min_{v \in C} \langle d_t, v \rangle$$

$$3: \gamma_t \leftarrow \text{StepSizeStrategy}(x_t, t, d_t, v_t)$$

Each component has predefined types (simplex LMO, agnostic step size...) and a way to define your own.

Example usage

$$\min_{x \in \Delta} \frac{1}{2} \|x - y\|^2$$

```
using FrankWolfe
using LinearAlgebra
n = 1000
y = randn(n)
function f(x)
    return 1/2 * norm(x - y)^2
end
function grad!(storage, x)
    # perform entrywise without temporary allocation
    @. storage = x - y
end
```

```
lmo = FrankWolfe.ProbabilitySimplexOracle(1.0)
x0 = FrankWolfe.compute_extreme_point(lmo, zeros(n))
xfinal, vfinal, primal_value, _ = FrankWolfe.frank_wolfe(
    f, grad!, lmo, x0,
    max_iterations=1000, epsilon=10^-8, # other options
)
```

FrankWolfe.jl, how and where?

Registered on the official Julia package registry:

```
using Pkg
Pkg.add("FrankWolfe")
```

```
using FrankWolfe
# good to go
```

Open-source license (MIT)

Available on <https://github.com/ZIB-IOL/FrankWolfe.jl>

Software paper: <https://arxiv.org/abs/2104.06675>

More complex feasible region, no closed-form solution?

Accepts problems defined through MathOptInterface / JuMP

Frank-Wolfe algorithms

FrankWolfe.jl: a Julia implementation

Frank-Wolfe and generalized self-concordance

TL;DR:

- Generalized self-concordance: useful class of functions, looser assumptions
- We show that Frank-Wolfe converges with the usual rate
- Highlight that a simple twist on $2/(t + 2)$ step size converges
- Also works great on computational experiments.

Article:

Simple steps are all you need: Frank-Wolfe and generalized self-concordant functions

<https://arxiv.org/abs/2105.13913>

Assumptions for FW vs. self-concordance

- σ -smoothness - Lipschitz-continuous gradient
- μ -strong convexity

$$\mu I \preceq \nabla^2 f(x) \preceq \sigma I$$

(M, ν) generalized self-concordance: $f \in C^3(\text{dom } f)$, closed convex with $\text{dom } f \subseteq \mathbb{R}^n$ open.

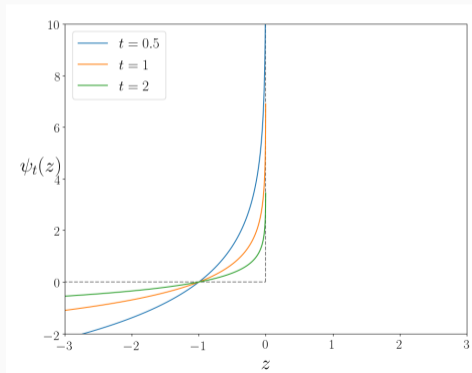
$$|\langle D^3 f(x)[w]u, u \rangle| \leq M \|u\|_{\nabla^2 f(x)}^2 \|w\|_{\nabla^2 f(x)}^{\nu-2} \|w\|_2^{3-\nu} \quad \forall x \in \text{dom } f, u, w \in \mathbb{R}^n$$

Univariate case:

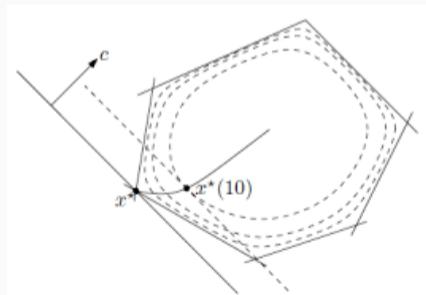
$$|\phi'''(t)| \leq M \phi''(t)^{\frac{\nu}{2}}$$

Examples?

- Logistic regression
- KL divergence between probability measures
- **Log-barrier functions:**



1



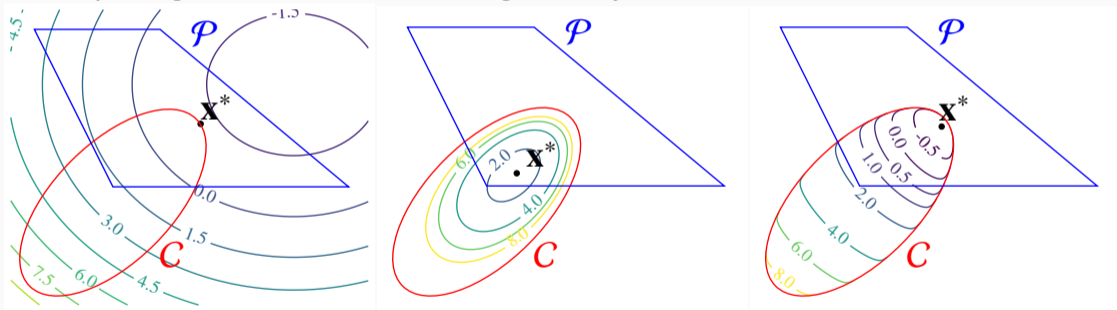
2

¹Source: <https://deepai.org/publication/log-barrier-constrained-cnns>

²Source: <http://www.stat.cmu.edu/~ryantibs/convexopt-S15>

Exploiting barrier functions

Improving constraint structure through the objective



- Focus on **second-order methods**
- (Generalized) self-concordant property designed for Newton-type algorithms [Sun, 2018, Sun and Tran-Dinh, 2019]
- First work on Frank-Wolfe but Hessian information required for step size [Dvurechensky et al., 2020, Liu et al., 2020].

Our contribution

Typical open loop step size strategy:

$$\gamma_t = \frac{2}{2+t}$$

Monotonous variant:

$$\gamma_t := \frac{2}{2+t}$$

while $x_t + \gamma_t(v_t - x_t) \notin \text{dom } f$ and $f(x_t) \geq f(x_t + \gamma_t(v_t - x_t))$: $\gamma_t = \frac{\gamma_t}{2}$

Converges with $O(1/t)$, using a **Domain Oracle**.

Also:

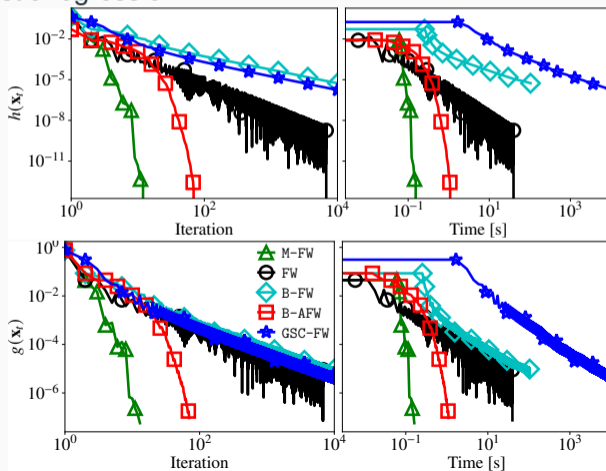
Improved convergence for uniformly convex sets

Convergence for Away-step FW.




Computational results



TL;DR: Away-step usually the fastest converging method, monotonous competitive, second-order too expensive and not much per-iteration progress.

Example on a logistic regression:



- Generalized self-concordance: alternative to strong convexity + smoothness requirements
- Frank-Wolfe algorithms Just WorkTM with such property
- FrankWolfe.jl: competitive toolbox for our and others' research
- Developing and offering ready-to-use algorithms for non-linear optimization with flexible constraints & objective.

-  Besançon, M., Carderera, A., and Pokutta, S. (2021).
FrankWolfe.jl: a high-performance and flexible toolbox for Frank-Wolfe algorithms and Conditional Gradients.
arXiv preprint arXiv:2104.06675.
-  Carderera, A., Besançon, M., and Pokutta, S. (2021).
Simple steps are all you need: Frank-Wolfe and generalized self-concordant functions.
arXiv preprint arXiv:2105.13913.
-  Dvurechensky, P., Safin, K., Shtern, S., and Staudigl, M. (2020).
Generalized self-concordant analysis of Frank-Wolfe algorithms.
arXiv preprint arXiv:2010.01009.

-  Liu, D., Cevher, V., and Tran-Dinh, Q. (2020).
A Newton Frank-Wolfe method for constrained self-concordant minimization.
arXiv preprint arXiv:2002.07003.
-  Sun, T. (2018).
Newton-type methods under generalized self-concordance and inexact oracles.
PhD thesis, University of North Carolina at Chapel Hill Graduate School.
-  Sun, T. and Tran-Dinh, Q. (2019).
Generalized self-concordant functions: a recipe for Newton-type methods.
Mathematical Programming, 178(1):145–213.