Algorithmic Trading with Reinforcement Learning

Second semester report

Leonardo Toffalini

Supervised by András Lukács

1 Introduction

During this semester we continued where we left off after my undergraduate thesis and first semester work. Given the length constraints of the present report, we only focus on defining the problem at hand and briefly mention some notable achievements.

Consider a modeled financial market wherein the price of a risky asset adheres to an adapted process S_t, where $t \in [0,T]$. In our case S_t is a fractional Brownian motion. The trader may trade at finite rates on the risky asset, though they incur a temporary nonlinear price impact as a consequence. The family of feasible strategies available to the trader is

$$S(T) := \left\{ \phi : \phi \text{ is a } \mathbb{R}\text{-valued optional process and } \int_0^T |\phi| \, \mathrm{d}u < \infty \text{ a.s.} \right\}.$$

The trader's initial asset position is represented by $z = (z^0, z^1)$, where z^0 is the number of units of the riskless asset, and z^1 represents the number of units of risky assets.

The number of units of the risky asset at time $t \in [0,T]$, after following the strategy ϕ is given by

$$X_t^1(\phi) := z^1 + \int_0^t \phi_u \, \mathrm{d}u.$$

The aggregate position in the riskless asset is defined in a comparable manner, albeit incorporating the effect of price impact. The trader incurs a superlinear penalty associated with the trading speed, as determined by parameters $\alpha > 1$ and $\lambda > 0$. The aggregate position in the riskless asset at time $t \in [0,T]$ is given by

$$X_t^0(\phi) := z^0 - \int_0^t \phi_u S_u \, \mathrm{d}u - \int_0^t \lambda |\phi_u|^\alpha \, \mathrm{d}u.$$

Let $\mathcal{A}(T)$ be the family of feasible strategies starting with zero initial capital, and with the final position composed exclusively of the riskless asset, and a well-defined notion of expected terminal riskless asset position, that is

$$\mathcal{A}(T) := \{\phi \in S(T) : X_T^1 = 0, \quad \mathbb{E}[X_T^0(\phi)_-] < \infty\},$$

where $x_- = -\min(x, 0)$.

The objective of our problem is to identify the strategy $\phi \in \mathcal{A}(T)$ that realizes maximal expected profits of the riskless asset.

It can be shown that there exists an optimal strategy for any time horizon T that achieves maximal returns [1]. It can also be shown that a simple contrarian strategy in the anti persistent case, and a momentum strategy in the persistent case with linear liquidation after $T/2$ steps achieves asymptotically optimal returns, that is of order $T^{H(1+\kappa)+1}$ [1], when $\kappa \to 1/(\alpha - 1)$.

The goal of this project, and that of my undergraduate thesis, is to compete with the analytical results by learning a strategy that compares in its expected returns. We will learn such a strategy using reinforcement learning (RL).

2 Previous work

2.1 Bsc

During my undergraduate thesis, we showed that with a standard PPO [2] algorithm we were able to outcompete the analytical strategy on expected returns for time horizons $T \leq 512$ by training a bespoke model for each tested time horizon, which is to say that we did not find a general strategy that worked for any time horizon. For time horizons greater than 512 the learned strategies were not able to outperform the simple analytical strategy on expected returns, which we attributed to the credit assignment problem becoming increasingly more difficult for longer horizons.

2.2 First semester

In the first semester of this project we kept the overall methodology with which we started initially, but reimplemented everything in a more performant manner achieving a roughly 3 orders of magnitude faster training. The other major component we changed was that we restricted the problem to finding only the first part of the strategy, and then we automatically liquidate the amassed position linearly, as it can be shown that no better liquidation schedule can be given in the present problem other than linear.

3 Current work

One of the major drivers of the current semester's work was to loosen the constraints of the reinforcement learning problem, of which the action space was the most drastic change.

Previously, the action space was set to $\mathcal{A} = \{-K, ..., 0, ..., K\}$, where K was the largest amount the agent could trade. The problem with this was that the agent only learnt two different types of trades, one sell order and one buy order. Essentially, the agent either bought a shares, or sold b shares, no in between. This phenomenon can be explained by the model architecture see Figure 1, as when the action space is discrete the agent learns a categorical distribution on the action space. Since the model does not know that output 2 represents a trade 1 share less than output 3, i.e. the model has no knowledge of the ordering of the output, it cannot make fine adjustments.

The original problem statement defines the action space to be $(-\infty, +\infty)$, however, in reinforcement learning there is a great gap in complexity for problems with discrete versus continuous action spaces, this is why in the beginning we restricted the problem to strategies restricted to trading fixed quantities. Clearly, the solution to the problem was to let the action space be continuous.

In the continuous case the policy architecture does not create a categorical distribution over $2K + 1$ actions, but instead it creates a normal distribution $N(\mu, \sigma^2)$, where the policy network

has a single output value: μ, and σ^2 is set to be a module level learnable parameter, independent of the input, see Figure 2.

The actual policy network architecture consists of three parts: observation encoder, LSTM brain, and action decoder. The encoder and decoder are of simple MLP design with 2-3 layers, and the inner LSTM is of a single layer and of 384 hidden layer size, mirroring the architecture used by OpenAI Five [3]. For the accurate representation of the policy network architecture see Figure 6 in the Appendix.

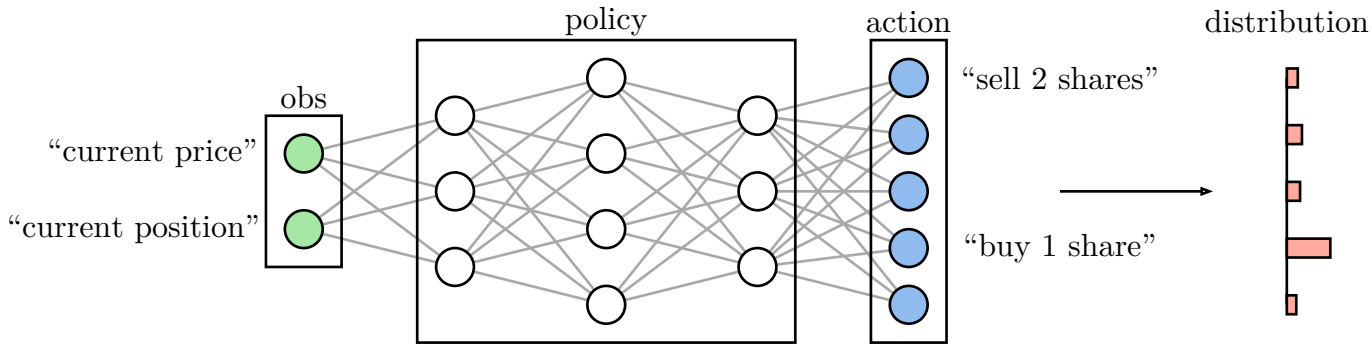

Figure 1: Schematic representation of the discrete action policy architecture.

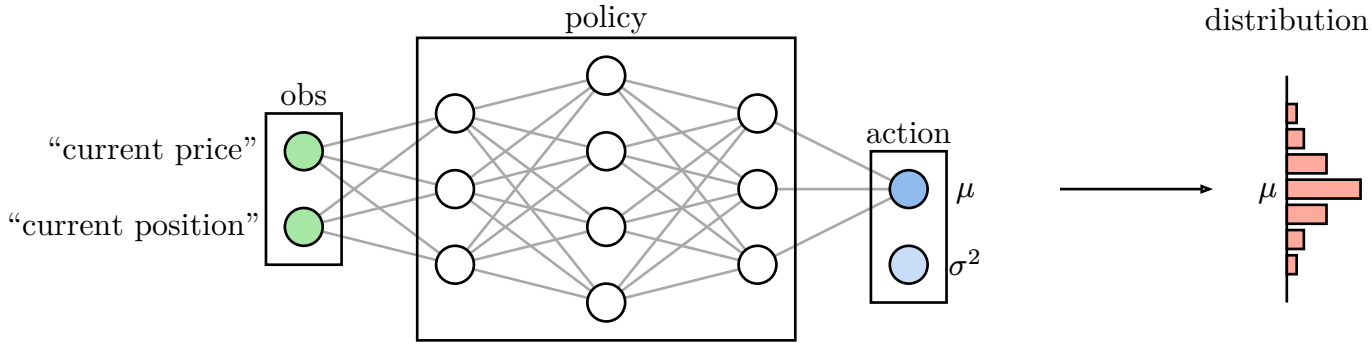

Figure 2: Schematic representation of the continuous action policy architecture.

The next biggest improvement that was implemented is that now the time horizon T is not fixed during training, but it is sampled from a distribution for each episode while training. The general approach of modifying the problem setting randomly in hopes of finding a more robust policy is a well studied approach called domain randomization [4]. For simplicity, the following distribution was used

$$T = T_{\min} + (T_{\max} - T_{\min})U^p, \quad \text{where } U \sim U[0, 1].$$

This distribution was chosen to be able to control which time horizons get more attention during training, with $p = 1$ we get a uniform distribution, with $p > 1$ we oversample shorter time horizons, while with $p < 1$ we oversample longer time horizons, as can be seen in Figure 3.

There are different reasons that motivate a biased distribution of the time horizon, one practical consideration is that if we were to simply train on a uniform distribution of time horizons, the endings of shorter horizons would be over represented in each batch as they terminate more often. The other reason to switch to a biased distribution, is that in the future we aim to employ curriculum learning, where we train the model on problems of increasing difficulty, by starting training on a distribution that oversamples shorter then longer horizons, thus shifting focus on more and more longer horizons.

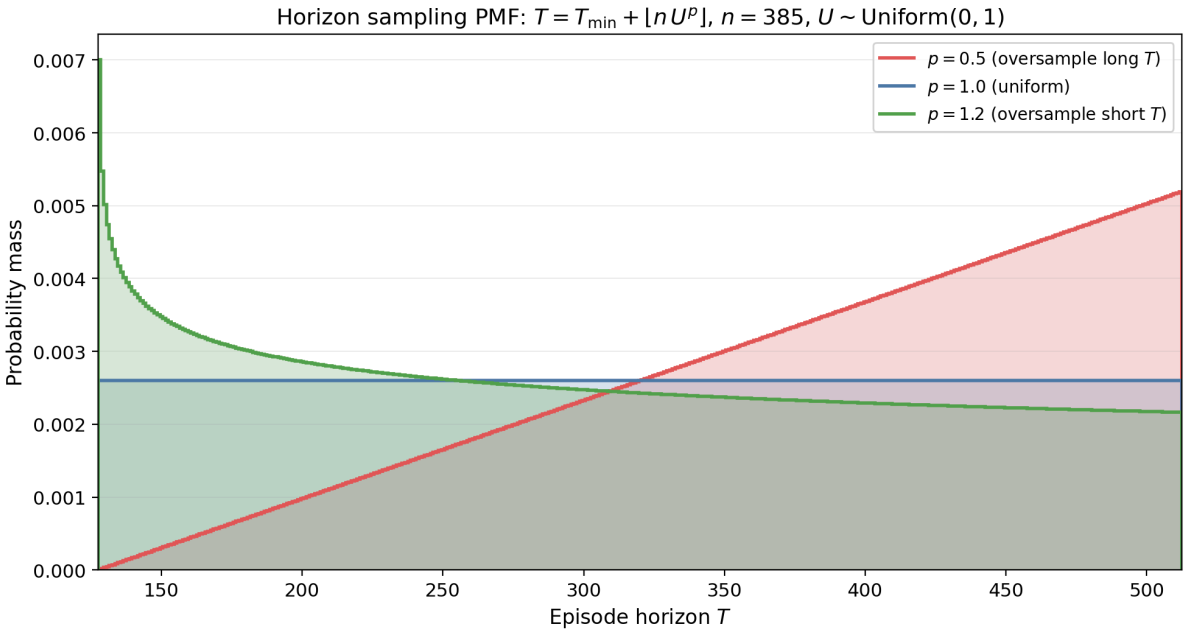

Figure 3: Distribution of time horizons with respect to p.

A key theoretical result motivating this project is that the returns of any admissible strategy are bounded by a market-dependent upper bound. More precisely,

$$\left|X_T^0(\phi)\right| \leq C \int_0^T |S_t|^{\alpha/(\alpha-1)} \, \mathrm{d}t = CQ(T),$$

where

$$C = \frac{\alpha - 1}{\alpha} \alpha^{\frac{1}{1-\alpha}} \lambda^{\frac{1}{1-\alpha}}.$$

Furthermore, the scaled contrarian strategy introduced in [1] is asymptotically optimal in the anti persistent case, in the sense that it achieves the same asymptotic growth rate as the market bound. This bound therefore serves as the natural benchmark against which we evaluate the learned policy.

The asymptotically optimal strategy introduced in [1] takes the form

$$\phi_t(T) := \operatorname{sgn}(S_t(H - 1/2))|S_t|^{1/(\alpha-1)} \qquad (t \in [0, T/2))$$

for the first half of the trading period, and linear liquidation for the second half $t \in [T/2, T]$.

Figure 4 compares the learned policy against the asymptotically optimal analytical strategy and the market bound. Although the policy was trained only on trajectories with T=256, it generalizes well to both shorter and substantially longer horizons, suggesting that the learned strategy is approximately homogeneous in time t and horizon T.

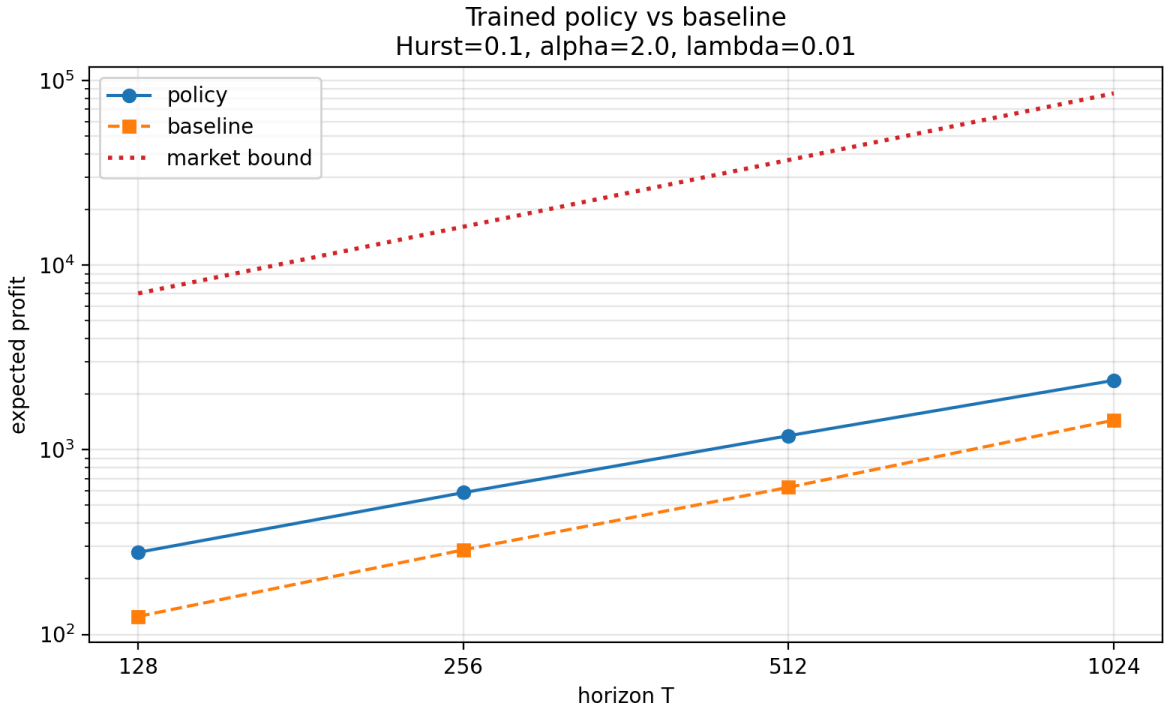

Figure 4: Learnt policy versus asymptotically optimal strategy versus market bound. Each dot represents the mean return of 500 rollouts.

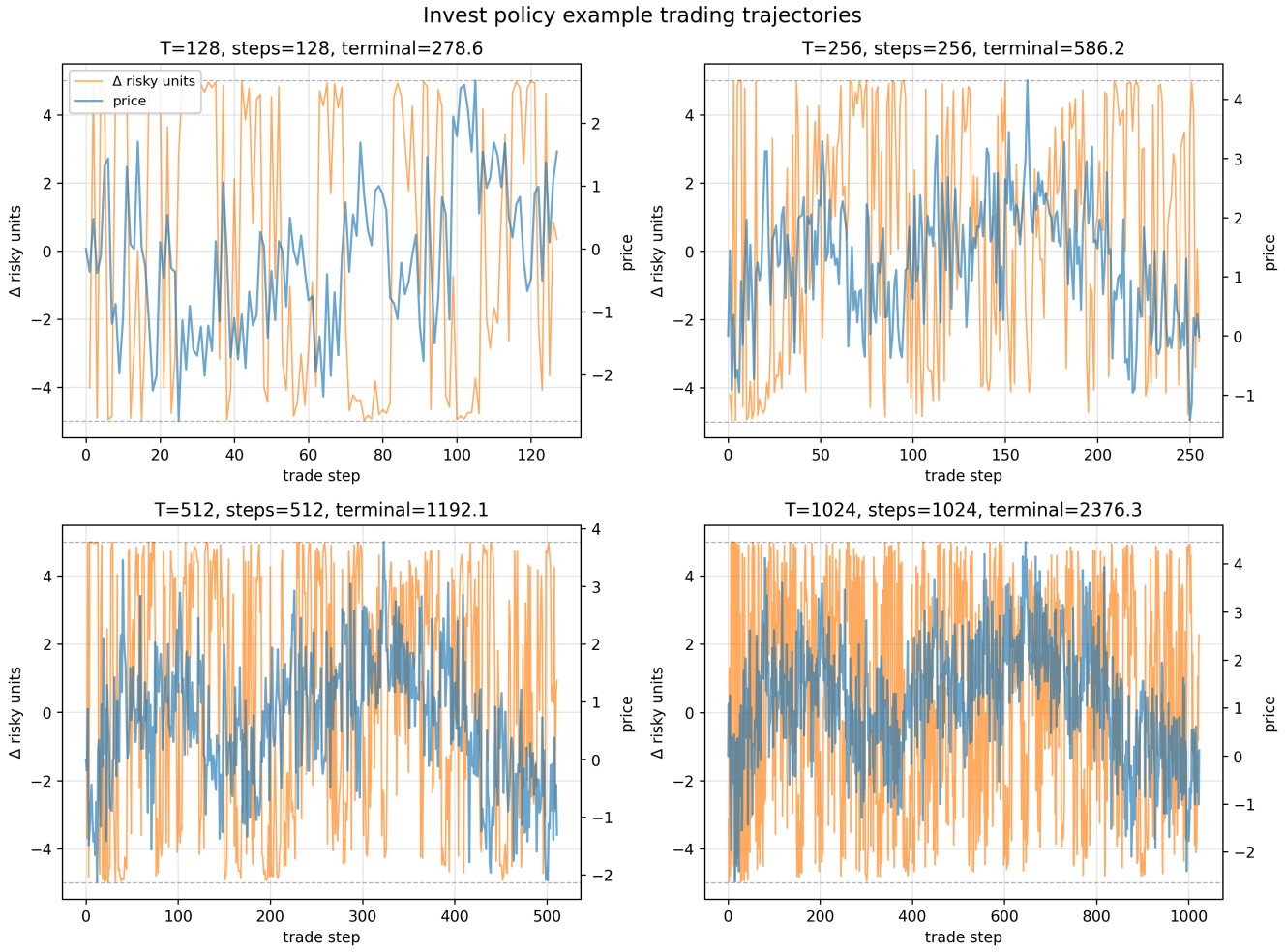

Figure 5: Example rollouts showing only the action – delta riskless units – and the fBm prices for increasing time horizons. The achieved return is noted in the subtitle of each subplot.

Appendix

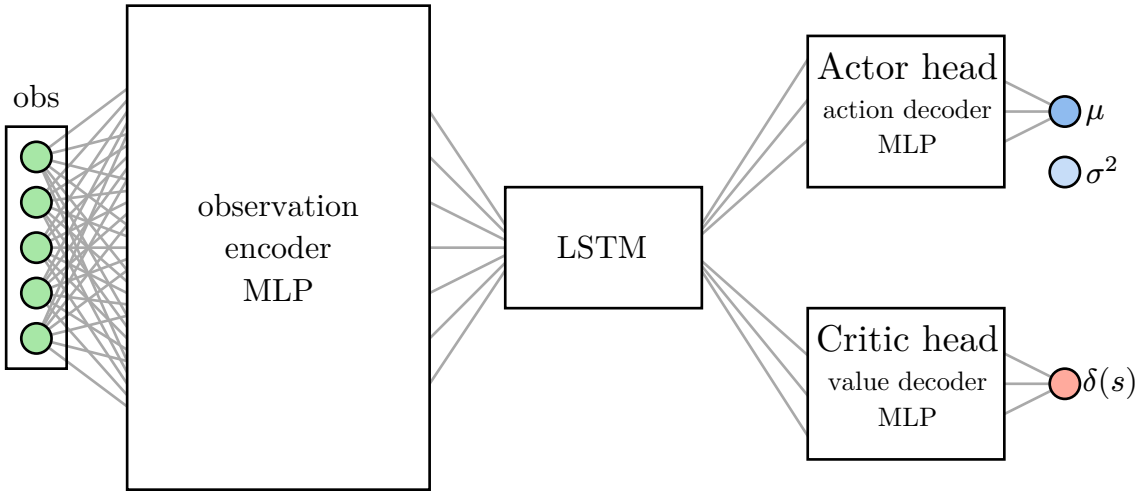

Figure 6: Policy network architecture consisting of an observation encoder, recurrent LSTM core, and actor-critic output heads.

Name	Value
rnn.hidden_size	384
adam_beta1	0.9951718674696215
adam_beta2	0.9413939175583763
adam_eps	0.0001
anneal_lr	“true”
batch_size	auto
bptt_horizon	64
checkpoint_interval	200
clip_coef	0.27054937558218256
ent_coef	0.20000000000000004
final_eval_passes	1
gae_lambda	0.9898748083815585
gamma	0.9382600900682131
learning_rate	0.001051989540888071

Name	Value
max_grad_norm	2.7495165717556604
max_logratio	12
max_minibatch_size	32768
minibatch_size	32768
prio_alpha	0.853334170486411
prio_beta0	0.8897892497407985
seed	42
target_kl	0.03
total_timesteps	100_000_000
update_epochs	1
vf_clip_coef	0.1
vf_coef	3.4981909434385705
vtrace_c_clip	0.1900890933017968
vtrace_rho_clip	1.2270571100490357

Table 1: List of hyperparameters used for training.

Bibliography

[1] P. Guasoni, Z. Nika, and M. Rásonyi, "Trading fractional Brownian motion," SIAM journal on financial mathematics, vol. 10, no. 3, pp. 769–789, 2019.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[3] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," arXiv preprint arXiv:1912.06680, 2019.

[4] A. Elsafi, "Evaluating Domain Randomization Techniques in DRL Agents: A Comparative Study of Normal, Randomized, and Non-Randomized Resets.," Computer Modeling in Engineering & Sciences (CMES), vol. 144, no. 2, 2025.

[5] S. Huang et al., "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms," Journal of Machine Learning Research, vol. 23, no. 274, pp. 1–18, 2022, [Online]. Available: http://jmlr.org/papers/v23/21-1342.html

[6] R. S. Sutton, A. G. Barto, and others, Reinforcement learning: An introduction, vol. 1. MIT press Cambridge, 1998.

[7] J. Gatheral, T. Jaisson, and M. Rosenbaum, "Volatility is rough," Quantitative finance, vol. 18, no. 6, pp. 933–949, 2018.

[8] J. Suarez, "PufferLib 2.0: Reinforcement Learning at 1M steps/s," Reinforcement Learning Journal, vol. 6, pp. 1378–1388, 2025.

[9] A. J. Fetterman et al., "Tune As You Scale: Hyperparameter Optimization For Compute Efficient Training," arXiv e-prints, 2023.

[10] P. Guasoni and M. Rásonyi, "Hedging, arbitrage and optimality with superlinear frictions," 2015.