Preprint October 28, 2014
To appear in Humanoids 2014, Madrid, Spain

ROCK* - Efficient Black-box Optimization for Policy Learning

Jemin Hwangbo[1,2], Christian Gehring[3,4], Hannes Sommer[3], Roland Siegwart[3] and Jonas Buchli[2]

***Abstract*— Robotic learning on real hardware requires an efficient algorithm which minimizes the number of trials needed to learn an optimal policy. Prolonged use of hardware causes wear and tear on the system and demands more attention from an operator. To this end, we present a novel black-box optimization algorithm, Reward Optimization with Compact Kernels and fast natural gradient regression (ROCK*). Our algorithm immediately updates knowledge after a single trial and is able to extrapolate in a controlled manner. These features make fast and safe learning on real hardware possible. We have evaluated our algorithm on two simulated reaching tasks of a 50 degree-of-freedom robot arm and on a hopping task of a real articulated legged system. ROCK* outperformed current state-of-the-art algorithms in all tasks by a factor of three or more.**

I. INTRODUCTION

The control of robots, in particular humanoid robots, presents grand challenges for control. Finding optimal or even feasible control policies is difficult, as modern robots are generally nonlinear and non-smooth dynamical systems with high-dimensional state and action spaces. A possible approach to this kind of control problem is reinforcement learning, which has become widely used in robotics [1]. In reinforcement learning, either a reward is maximized or a cost is minimized by exploring different control strategies. The execution of a policy in a task, a so-called rollout, is safe and cheap in simulation, but costly in experiments because of wear and tear on the robot. For this reason fast-converging algorithms are highly desirable for learning new motor skills.

Our goal is to learn variable impedance control for articulated robots, such as robots with arms and legs. To overcome some of the shortcomings of existing algorithms we derived a novel algorithm that allows us to optimize control policies on real hardware with a minimum number of rollouts. We take a black-box optimization approach to reinforcement learning and demonstrate the resulting performance of the novel algorithm on reaching-tasks of a simulated 50 degree of freedom robotic arm and the optimized variable impedance policy of the real robotic leg.

Classic reinforcement learning algorithms such as Q-Learning [2] require to learn action-value function for all discretized states, which is intractable for continuous-time and high-dimensional systems. To avoid having to discretize the high-dimensional state and action spaces, parameterized policies, employing Fourier series or splines, are commonly used to reduce the number of optimization variables. Some of the state-of-the-art reinforcement learning algorithms, such as PI^2 [3], its variant PI^2-CMA [4], and eNAC [5], utilize this strategy. Discrete reward task learning [6] and optimizing jumping tasks for a single robotic leg in experiments [7] was demonstrated using PI^2. The disadvantage of these so-called gray-box optimization algorithms is that they generally use a fixed-policy representation and do not generalize well to other problems. In addition, a recent study [8] has shown that they do not necessarily outperform the state-of-the-art black-box optimization algorithm CMA-ES [9]. CMA-ES was successfully used in realistic scenarios such as tuning the control parameters for a quadrupedal locomotion controller to enable various dynamic gaits [10] and optimizing reference trajectories to manipulate a heavy object with a robot arm mounted on a quadruped [11].

Existing black-box optimization methods which implement local gradient search such as FDSA [12], SPSA [13], IW-PGPE [14] and REINFORCE [15] explicitly approximate the gradient of the objective. They however require many rollouts, especially in presence of system noise [16], and often show convergence problems when the objective, the cost function, is non-differentiable.

Some black-box optimization algorithms such as CMA-ES, AMalGaM [17] and CEM [18], based on random Gaussian sampling of the policy parameters and the subsequent reward weighted averaging (RWA) of the sampled policy parameter sets, avoid the latter problem by following an implicit gradient. Nevertheless, these algorithms follow a local gradient and can thus get trapped at a local optimum. An approach to find the global optimum is GP-UCB [19], which maximizes the combination of expected reward and information gain. This method is however not well suited for robotic applications since the algorithm does not scale well for high dimensions and starts searching for policies on the boundaries of the action space, which may be harmful for the robot or at least sophisticated safety and sanity checks have to be put in place.

To the best of our knowledge, local optimization algorithms are best suited for optimizing high dimensional control policies, but current RWA-based algorithms have three shortcomings that lead to a unnecessarily high number of rollouts until convergence. First, the algorithms update their knowledge only after a 'batch' of rollouts instead of updating immediately after each rollout. Second, only the best seen information (elitism) is considered. Third, only a convex combination of these best policies (reward weighted averaging) are picked to generate new rollouts. Our new black-box optimization algorithm, named **ROCK*** (**R**eward

[1] jhwangbo@ethz.ch – The initial research to this study has been done while the first author was at the Agile & Dexterous Robotics Lab
[2] Agile & Dexterous Robotics Lab (ADRL), ETH Zurich, Switzerland
[3] Autonomous Systems Lab (ASL), ETH Zurich, Switzerland
[4] Disney Research Zurich, Switzerland

Optimization with **C**ompact **K**ernels and fa**st** n**a**tural gradient **r**egression), does not suffer from these drawbacks. The approach is based on Gaussian kernel regression of the cost function and the subsequent minimization of the regressed function using natural gradient descent.

The following Sect. II introduces the ROCK* algorithm. Section III presents the results of the optimization of very high dimensional simulation tasks. Section IV shows how useful the algorithm is to find variable impedance control policies for hopping with a robotic leg and briefly discusses the resulting strategies. The paper concludes with a short discussion in Sect. V.

II. ROCK* ALGORITHM

Our objective is to find a control policy parameter vector that minimizes the expected cost,

$$J = \underbrace{\mathrm{E}[h(\boldsymbol{x}(t_e)) + \int_0^{t_e} g(\boldsymbol{x}(t), \boldsymbol{u}(t, \boldsymbol{\theta}))dt]}_{R}, \tag{1}$$

where h is a terminal cost, g is an immediate cost, $\boldsymbol{x}$ is the system state, $\boldsymbol{u}$ is a control input and $\boldsymbol{\theta}$ is a policy parameter vector. In a black-box optimization setting, we only have access to the previously executed policy parameter vectors $\boldsymbol{\theta}_i$ and their corresponding cost values r_i. To find the next policy to try, ROCK* performs kernel regression on J using $\boldsymbol{\theta_i}$ and r_i and subsequently finds the minimum using natural gradient descent.

A. Kernel Regression

Regression is used in ROCK* not only to estimate the underlying function shape, but also to penalize unvisited regions such that too high exploration is prohibited. To cope with our needs, we designed the following regression form:

$$\begin{aligned}
\mathrm{E}[R|\boldsymbol{\theta}] &\approxeq \frac{1}{\tilde{L}(\boldsymbol{\theta})}\left[\sum_{i=1}^{N} r_i e^{-\frac{1}{2}(\boldsymbol{\theta}_i-\boldsymbol{\theta})^T\ (2\boldsymbol{\Sigma_\epsilon})^{-1}(\boldsymbol{\theta}_i-\boldsymbol{\theta})} + C\right], \\
\tilde{L}(\boldsymbol{\theta}) &= \sum_{i=1}^{N} e^{-\frac{1}{2}(\boldsymbol{\theta}_i-\boldsymbol{\theta})^T\ (2\boldsymbol{\Sigma_\epsilon})^{-1}(\boldsymbol{\theta}_i-\boldsymbol{\theta})} + 1.
\end{aligned} \tag{2}$$

where $\boldsymbol{\Sigma_\epsilon}$ is the covariance of a kernel function, and C is an auxiliary cost. C is calculated as,

$$C = \frac{1}{|I|}\sum_{i\in I} r_i,\ I := \{i|(\boldsymbol{\theta}^* - \boldsymbol{\theta}_i)^T {\boldsymbol{\Sigma_\epsilon}}^{-1}(\boldsymbol{\theta}^* - \boldsymbol{\theta}_i) < \lambda_{\mathrm{MD}}^2\}. \tag{3}$$

where λ_{MD} is a threshold that defines the search radius and $\boldsymbol{\theta}^*$ is the previously estimated optimum policy vector. This equation is very similar to a linear smoother with Gaussian kernels, commonly called as Nadarya-Watson regression. However, the auxiliary cost C and its probability density gives this regression form a unique extrapolation characteristic. Note that Gaussian smoothing guarantees that the algorithm does not get trapped in artificial minima which is not present in the underlying function. The auxiliary cost C generates an artificial minimum but accumulation of samples eventually leads to exploration since the effect of C diminishes as the samples accumulate. This minimum is usually formed outside of the convex hull of the sampled parameter vectors in the direction of the parameter vectors with the lower costs. This is always true in a two point case given that the two samples have different costs. When the samples are scarce locally, it behaves similarly. If there are many samples in a small region, the algorithm behaves similar to Gaussian smoothing and the behavior is dependent on the underlying function. The resulting algorithm is illustrated in Fig. 1. Fig. 1a) shows an original function J and a function $\tilde{J}$ regressed with samples from J. With many samples concentrated in a small region, Eq. 2 behaves like a linear smoother. This smoothing behavior is very effective near an optimum, where good convergence behavior is essential. Fig. 1c) shows an example of learning progress of ROCK*. When there are first two samples, which are marked with red dotted circles, the minimum is formed at the red dot. It shows the extrapolation property of ROCK*. It is clearly different from batch-based update strategy illustrated in Fig. 1b). The rate of extrapolation can be controlled by adjusting the kernel size as,

$$\boldsymbol{\Sigma_\epsilon} = -\frac{1}{2\ln(\lambda)} P(0.95, n)\boldsymbol{\Sigma}, \tag{4}$$

where P is the inverse of the Chi-squared cumulative distribution, and $\lambda \in (0, 1)$ is a tuning parameter. $\boldsymbol{\Sigma}$ is a exploration covariance which is used to sample the next policy parameter to try, as it is done in many other black-box optimization algorithms such as CMA-ES. Equation 4 defines a hyperellipsoid that contains 95% of the exploration possibilities and then selects the noise covariance, such that the probability density of the noise distribution at the boundary of this hyperellipsoid is equal to λ. The resulting extrapolation rate becomes independent of the number of dimensions of the policy parameter vector and, therefore, removes the need for an operator to hand tune it for different dimensions. The extrapolation rate increases with λ and we recommend to adjust it to a value between 0.4 and 0.6, which gives a moderate extrapolation rate.

B. Natural Gradient Descent

The next step in the algorithm is to find a minimum of Eq. 2 using natural gradient descent. Natural gradient descent [20] ensures fast convergence to a minimum given an appropriate metric. Assuming that our function is locally in a shape of the Gaussian kernel used in the regression, we choose $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1}$ as our metric tensor. Then the gradient descent equation can be written as,

$$\begin{aligned}
&\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \gamma \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1}(\nabla_{\boldsymbol{\theta}}\tilde{J})^T, \quad \text{where} \\
&\nabla_{\boldsymbol{\theta}}\tilde{J}(\boldsymbol{\theta}) = \frac{\sum_{i=1}^{N} r_i e^{-\frac{1}{2}\,\mathrm{MD}_i^2}\Gamma\tilde{L} - \sum_{i=1}^{N} e^{-\frac{1}{2}\,\mathrm{MD}_i^2}\Gamma\Lambda}{\tilde{L}^2}, \\
&\Gamma = \frac{1}{2}(\boldsymbol{\theta}_i - \boldsymbol{\theta})\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1}, \quad \Lambda = \sum_{i=1}^{N} r_i e^{-\frac{1}{2}\,\mathrm{MD}_i^2} + C, \\
&\mathrm{MD}_i^2 = (\boldsymbol{\theta}_i - \boldsymbol{\theta})^T\ (2\boldsymbol{\Sigma_\epsilon})^{-1}(\boldsymbol{\theta}_i - \boldsymbol{\theta}).
\end{aligned} \tag{5}$$

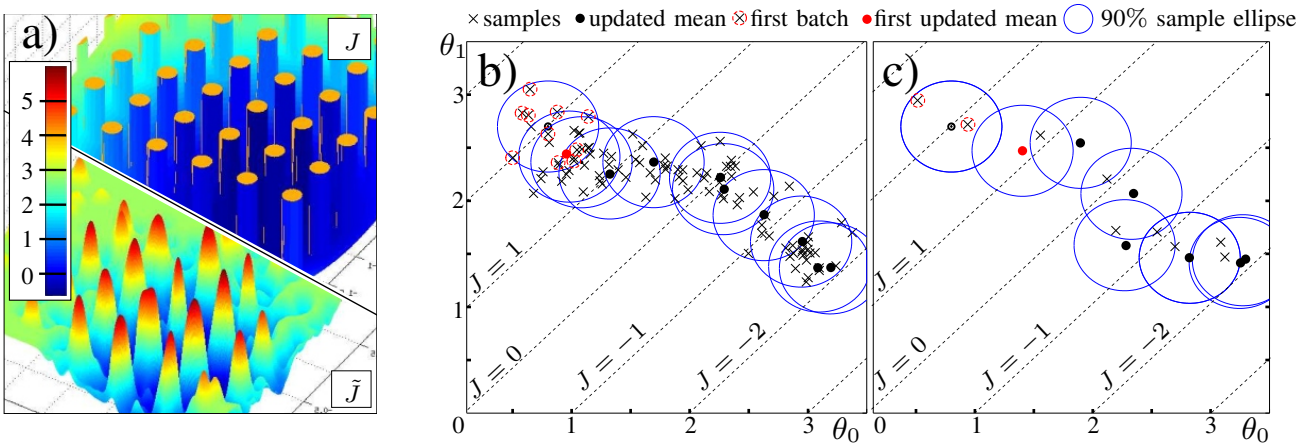

Fig. 1: a) shows an artificially designed cost function J and the estimated cost $\tilde{J}$. b-c) shows the advantage of the extrapolation property of the ROCK* (c) algorithm compared to a typical reward-weighted averaging algorithm (b).

where γ is a step size. γ is not task-specific and can be tuned automatically. From our locally approximated function $Ae^{-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\Sigma}_\epsilon^{-1}\boldsymbol{\theta}}$, we aim to cover half of the distance to the optimum in every update. The height of the Gaussian is approximated by the height of the highest kernel, $r_{min} - C$, where r_{min} is the lowest cost observed in rollouts. This gives: $\gamma = \frac{1}{2(C-r_{min})}$.

Euclidean distance is not a sufficient termination criteria since no scale is given to the algorithm. We chose to use Mahalanobis distance of the of the update vector, $\Delta\boldsymbol{\theta} := \gamma\boldsymbol{\Sigma}_\epsilon^{-1}(\nabla_{\boldsymbol{\theta}}\tilde{J})^T$, which is $\sqrt{\Delta\boldsymbol{\theta}^T\boldsymbol{\Sigma}_\epsilon^{-1}\Delta\boldsymbol{\theta}}$ instead.

Algorithm 1 ROCK*

Strategic Parameters
c_c, c_{cov}, ξ, κ, ζ and λ
Given
initial $\boldsymbol{\theta}^*$, exploration covariance $\boldsymbol{\Sigma}$,
$\vartheta \leftarrow -\frac{1}{2\ln(\lambda)}P(0.95, n)$, $\boldsymbol{\Sigma}_\epsilon \leftarrow \vartheta\boldsymbol{\Sigma}$, $\boldsymbol{P}_c \leftarrow \mathbf{0}$
$\boldsymbol{\Sigma}_u \leftarrow \boldsymbol{\Sigma}$, $\sigma \leftarrow 1$, $\psi \leftarrow det(\boldsymbol{\Sigma})$
repeat
  Sample a policy parameter vector, $\boldsymbol{\theta}_N \sim \mathcal{N}(\boldsymbol{\theta}^*, \boldsymbol{\Sigma})$
  Create a rollout $\boldsymbol{\theta}_N$ and evaluate $r_N = R(\boldsymbol{\theta}_N, \boldsymbol{w})$
  Update C (cf. Eq. 3)
  $\Delta\boldsymbol{\theta} \leftarrow \arg\min_{\boldsymbol{\theta}} \tilde{J}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*$ (cf. Eq. 2, Eq. 5)
  $\boldsymbol{\theta}^* \leftarrow \arg\min_{\boldsymbol{\theta}} \tilde{J}(\boldsymbol{\theta})$
  if $\sqrt{\Delta\boldsymbol{\theta}^T\boldsymbol{\Sigma}^{-1}\Delta\boldsymbol{\theta}} < \kappa\hat{\chi}_n$ **then**
    $\boldsymbol{P}_c \leftarrow (1 - c_c)\boldsymbol{P}_c + c_c\Delta\boldsymbol{\theta}/\sigma$
    $\boldsymbol{\Sigma}_u \leftarrow (1 - c_{cov})\boldsymbol{\Sigma}_u + c_{cov}\boldsymbol{P}_c\boldsymbol{P}_c^T$
    $\boldsymbol{\Sigma}_u \leftarrow (\psi/det(\boldsymbol{\Sigma}_u))^{1/n}\boldsymbol{\Sigma}_u$
  end if
  if $r_N > r_{N-1}$ **then**
    $\sigma \leftarrow \frac{1}{\zeta^\xi}\sigma$
  else
    $\sigma \leftarrow \zeta\sigma$
  end if
  $\boldsymbol{\Sigma} \leftarrow \sigma^2\boldsymbol{\Sigma}_u$, $\boldsymbol{\Sigma}_\epsilon \leftarrow \vartheta\boldsymbol{\Sigma}$
until convergence of $\tilde{J}(\boldsymbol{\theta}^*)$

C. Algorithm

Combining these regression and natural gradient descent techniques, we have designed a new black-box optimization algorithm, ROCK*. Algorithm 1 shows the pseudocode of the algorithm.

The algorithm starts by sampling two policy parameter sets from a given distribution. Then we use the proposed regression method to build an expected cost map and find a corresponding minimum. The minimum becomes the mean of the next distribution and exploration covariance and the kernel covariance are adjusted. To keep the same extrapolation rate, the two covariances need to be adjusted with the same factor. We have chosen a CMA-ES style covariance update method since it is known to be robust and efficient. The details on the adaptation method can be found in [9]. The details of implementation of ROCK* and its default strategic parameters can be found in our MATLAB code [21].

In the following sections, we present the testing results of ROCK* on various optimization tasks including a hopping task with a real robot.

III. Optimization in Simulation

In this section, we compare the solution and convergence rate of ROCK* with CMA-ES [9], PI^2 [3], and PI^{BB} [8] using the simulation of a 50 degrees-of-freedom robot arm from [3]. All tuning parameters of the individual algorithms were hand-tuned for every task individually to give optimal performance. The exploration covariance was found to be a very critical parameter while the other parameters have very small impact on the learning speed.

The first task investigated is the high-dimensional via-point task [3]. The robot's arm is initially lying on the x-axis and should reach the y-axis within 0.5 s while crossing the via-point G at time $t = 0.3$ s, as drawn in gray using a stroboscopic effect in Fig. 2a). The objective is to find the desired joint angles which minimizes the cost function J_1, as defined in Fig. 2c) and motivated by [3] and [8]. The desired joint angle trajectories are parameterized using Dynamical

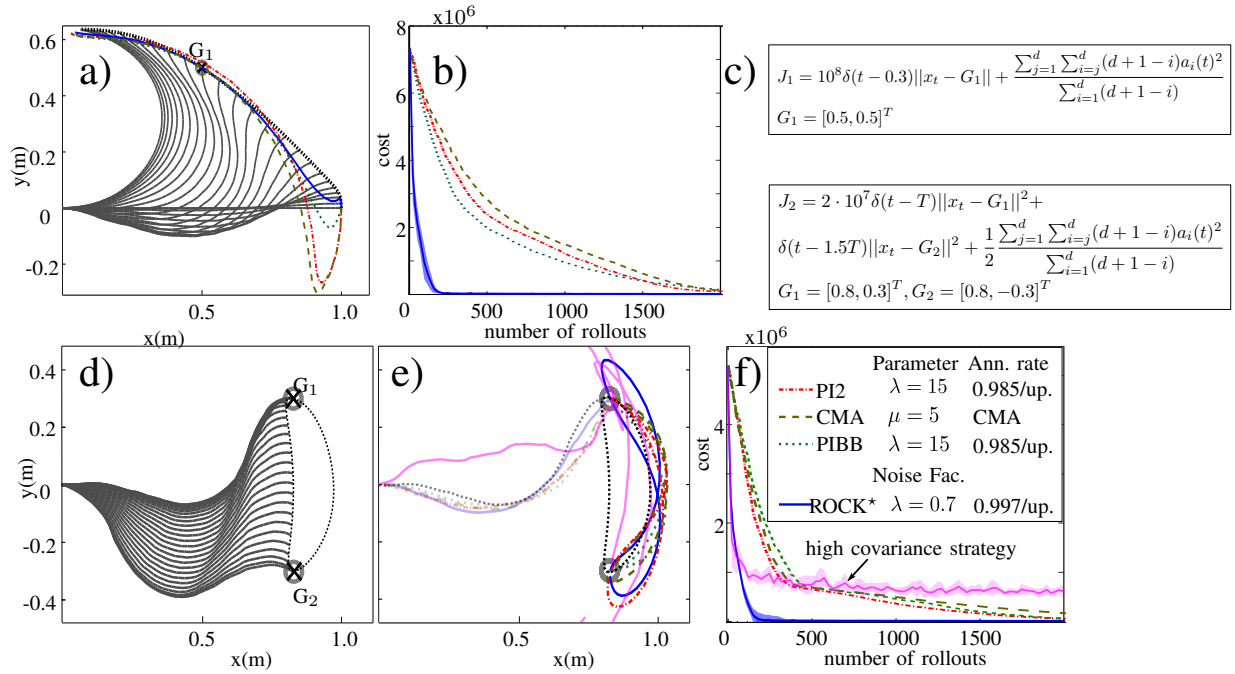

Fig. 2: Two tasks using a 50 DOF robot arm are presented. a) and b) show the learned end-effector trajectories and learning curves of the reaching task, respectively. e)-f) show those of the rhythmic task. The learning curves are averaged over 10 trials. The cost functions penalize the actuation cost and the distance to the via-points G at predefined time as shown in c).

Movement Primitives (DMP) [22] with 10 basis functions, resulting in a total of 500 policy parameters.

Fig. 2a) illustrates the end-effector trajectories of the best results of the different algorithms of 10 independent learning sessions each having 3,000 rollouts. The trajectories of the other algorithms conform to the motions published in [3].

TABLE I: The lowest costs for simulated arm tasks

Task	ROCK*	CMA-ES	PI^{BB}	PI^2
Reaching	21,765	34,892	42,834	68,419
Rhythmic	16,667	27,345	21,566	32,551

The lowest costs after 3,000 rollouts averaged over 10 trials are listed in Tab. I. After 20,000 rollouts, ROCK* finds the dashed solution shown in Fig. 2a), which has an even lower cost of 13,428. Fig. 2b) shows the learning curves of the algorithms averaged over 10 trials. While the other algorithms perform very similar to each other, ROCK* converges much faster.

The second task is a new rhythmic via-point task motivated by the challenges in legged locomotion. The goal is to drive the end-effector rhythmically through the two via-points G_1 and G_2 with a period of $T = 1\,\mathrm{s}$ as depicted in Fig. 2d). We modify the cost function as shown by J_2 in Fig. 2c), and use a Rhythmic Movement Primitive (RMP) [22] for each joint trajectory. The RMP is essentially a mixture of Gaussian kernels that are repeated with the predefined period T. Choosing 8 kernels per joint results in a total of 400 policy parameters.

Fig. 2e) visualizes the end-effector movements after 6,000 rollouts. Tab. I summarizes the final costs obtained by each algorithm. We found a near-optimal solution using ROCK* algorithm after 60,000 rollouts that leads to a minimal motion of the end-effector with a corresponding cost of only 3,110, visualized by the dashed black curves in Fig. 2e). The learning curves averaged over 10 learning sessions depicted in Fig. 2f) demonstrate that ROCK* also converges much faster than the other algorithms for this task.

The solid magenta curve in Fig. 2f) was generated using CMA-ES with a standard deviation 10 times bigger than that of the previous setup. This result shows the fundamental difference between large variance and smart extrapolation strategies. With a large variance strategy, the initial learning is as fast as ROCK* but fails to converge near the optimum. In high dimension, the chance of sampling a policy parameter vector which results in a low cost severely diminishes if the exploration covariance is increased, whereas the benefit of using an increased step size is only linear to the standard deviation. CMA-ES can auto-tune the size of the covariance but such adjustment in high dimension requires many rollouts. In addition, large standard deviation can initially generate many dangerous and impractical policies, whereas ROCK* only extrapolates in the region favored by the low cost samples.

IV. Optimization on Hardware

To evaluate the algorithm on real hardware, we used ROCK* to learn a variable impedance policy for robotic leg, ScarlETH [23]. The objective is to reject unperceived ground height perturbations. ScarlETH is a single legged planar hopping system with torque control capability, which allows impedance control at the joints. The main body has two degrees of freedom (x, z) but the horizontal degree

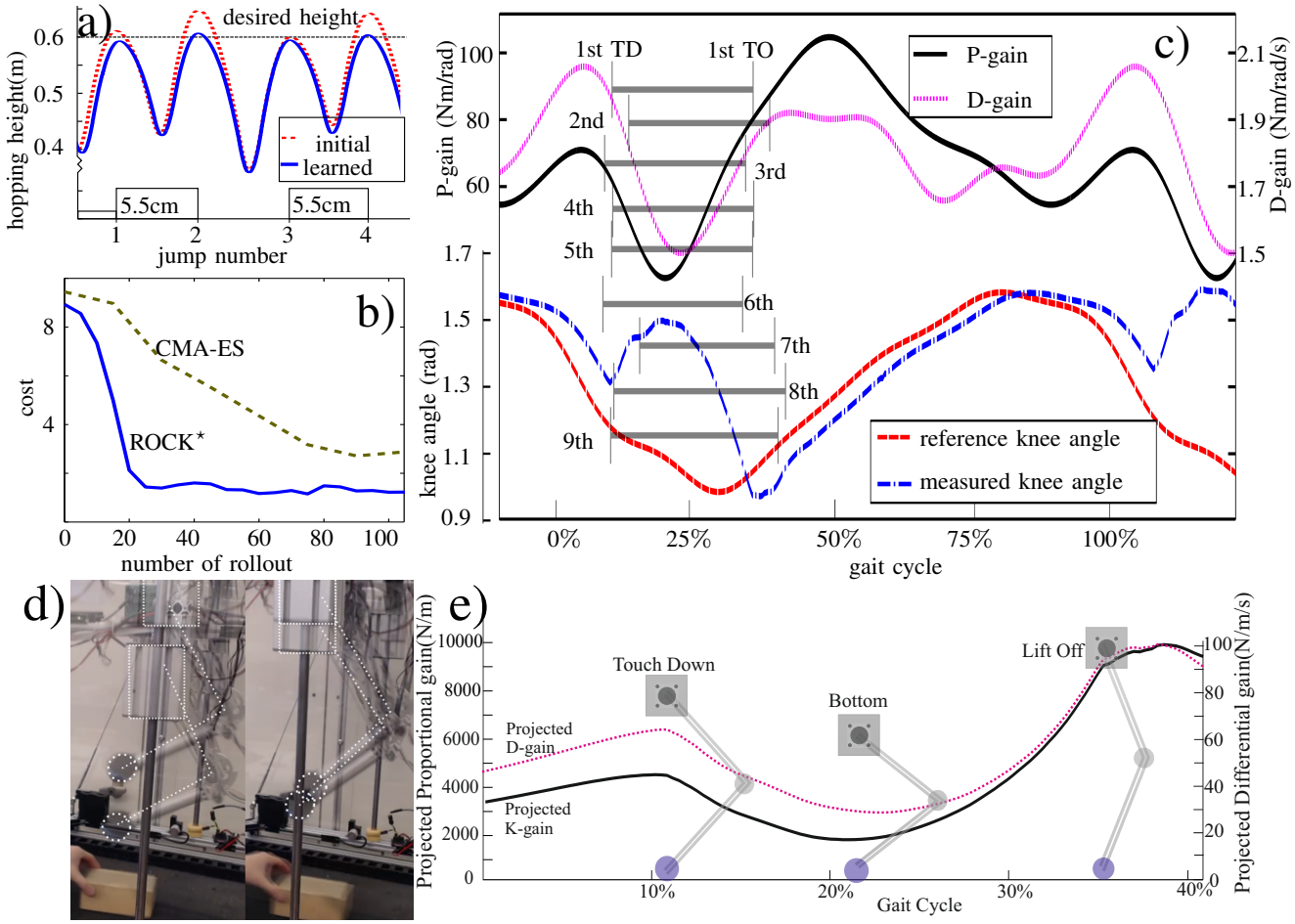

Fig. 3: Robotic leg learns variable impedance as it hops over unperceived objects. a) shows the part of the trajectories from initial and learned policy, b) shows the learning curves, c) shows the learned policy, i.e. control gains, knee trajectory and touch down (TD) and take off (TO) time and d) shows maximum and minimum apex height from the initial policy (left) and the learned policy (right) e) shows the impedance projected to the task space.

(x) was constrained mechanically for this experiment. The system is composed of three links, the main body, the thigh and the shank, and the two rotational joints located at the hip and the knee. Therefore ScarlETH has a total of 3 degrees of freedom for this experiment.

The controller used in the experiment is a PD-controller:

$$\boldsymbol{\tau} = k_p(\boldsymbol{x}_d - \boldsymbol{x}) + k_d(\dot{\boldsymbol{x}}_{\boldsymbol{d}} - \dot{\boldsymbol{x}}) \tag{6}$$

where k_p and k_d represent scalar proportional/differential gains, $\boldsymbol{x}$ represents the joint angles (i.e. knee and hip angles) and $\boldsymbol{x}_d$ represents reference trajectory. We used periodic policies for knee reference trajectory x_{d1} and for the two gains k_p and k_d, which were all parameterized with 10 periodically recurring Gaussian kernels. Since we are performing stationary hopping, the hip reference trajectory x_{d2} is constrained by the leg configuration given x_{d1} and is not part of control parameterization. We include the period of hopping in the parameter vector and the parameter vector becomes 31-dimensional.

The objective of the task is to reject the effect of unanticipated varying ground height during periodic hopping without explicitly perceiving the perturbation at the leg nor the position (height) of the main body. The ground height is varied from 0 cm to 1.5 cm and 5.5 cm by randomly sliding two different wooden blocks underneath the foot during the flight phase. The cost function penalizes deviations from the predefined apex height (0.6 m) and the number of hops, energy consumption and any command over the torque and joint limits. The initial policy was found from optimization in simulation with constant control gains.

Fig. 3a) illustrates the performance of the learned variable impedance policy with the body trajectory. While the apex height from initial policy fluctuates by about 5 cm in the presence of 5.5 cm block, the learned policy successfully minimizes this fluctuation to about 1 cm. The apex height stability is also shown in Fig. 3d), where two still shots at maximum and minimum apex heights are overlapping each other. Again, the learned policy shows significantly less variation in apex height. The source of this intrinsic disturbance rejection property can be found in Fig. 3c), which shows optimized time-varying gains and knee reference/measured trajectories. During the contact phase, marked by horizontal gray bars, the impedance of knee drops quickly. If the touch down is too early, the leg becomes more compliant during thrust phase and the impulse on the foot is reduced. Consequently, the hopping height is lowered which counteracts

the height gained from the wooden block. The impedance change is more dramatic when it is projected to the foot as shown in Fig. 3e). It becomes the lowest at the bottom phase and increases steeply thereafter. The articulation of the knee joint clearly amplifies the reduction in impedance. We can also interpret it that the articulated joint already has an intrinsic perturbation rejection property. As the knee bends, the foot impedance is lowered which can counteract ground perturbations and increase the stability of the system.

ROCK* successfully learned these dynamics of hopping only from cost feedback. CMA-ES was able to learn sufficient policy for stable hopping as shown in Fig. 3b) but with significantly more rollouts. Video which illustrates the performance of the learned impedance controller can be found at: http://y2u.be/23YaSzVDxJo

V. CONCLUSIONS

To achieve fast and safe learning in high dimensional, nonlinear, nonsmooth stochastic systems we developed a new parameterized policy optimization algorithm, ROCK*. Like other black-box optimization algorithms, ROCK* does not use any knowledge of the system dynamics or the tasks. It only explores and analyzes the relationship between the input parameters and their corresponding costs to find an optimal policy. This makes it simple for the algorithm to be implemented on complex systems. ROCK* operates on the following principles: immediate update of the current optimal estimate, controlled extrapolation and incorporation of all samples to average out the stochastic disturbances. These principles led us to a safe and efficient regression-style algorithm that performs well on real hardware tasks.

We have tested our algorithm in complex systems such as the 50 degree of freedom robot arm simulation and the real single legged hopping system, ScarlETH. For the two robot arm tasks, ROCK* successfully learned 500-dimensional and 400-dimensional policies to obtain very natural trajectories. The convergence speed of ROCK* was nearly an order of magnitude faster than the current algorithms and the learned policy manifested much lower costs than those from other algorithms. We also have validated that our algorithm can be used for optimizing rhythmic tasks by using it to optimize the hopping controller on our single legged robotic system. We parameterized the impedance and the trajectories with recurring Gaussian functions and optimized the policy with ROCK*. ROCK* successfully found that using a low impedance during the contact phase is the key in rejecting the effect of ground height disturbance. ROCK* was able to do this in significantly fewer rollouts than CMA-ES.

We plan to further investigate the performance ROCK* on quadruped robots such as HyQ[24] and StarlETH[25].

ACKNOWLEDGMENT

This research has been funded partially through a Swiss National Science Foundation Professorship award to Jonas Buchli and the NCCR Robotics.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[3] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, Dec. 2010.

[4] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *CoRR*, vol. abs/1206.4621, 2012.

[5] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7, pp. 1180–1190, 2008.

[6] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *Int. J. Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[7] P. Fankhauser, M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Reinforcement learning of single legged locomotion," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 188–193.

[8] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.

[9] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[10] C. Gehring, S. Coros, M. Hutter, M. Bloesch, H. M. A., and R. Siegwart, "Towards Automatic Discovery of Agile Gaits for Quadrupedal Robots," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[11] M. P. Murphy, B. Stephens, Y. Abe, and A. Rizzi, "High degree-of-freedom dynamic manipulation," in *Proceedings SPIE*, vol. 8387, 2012.

[12] J. Kiefer and J. Wolfowitz, "Stochastic estimation of a regression function," *Ann. Math. Stat.*, vol. 23, pp. 426–466, 1952.

[13] J. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins APL Technical Digest*, vol. 19, no. 4, 1998.

[14] T. Zhao, H. Hachiya, V. Tangkaratt, J. Morimoto, and M. Sugiyama, "Efficient sample reuse in policy gradients with parameter-based exploration," *Neural computation*, vol. 25, no. 6, pp. 1512–1547, 2013.

[15] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*, ser. The Springer International Series in Engineering and Computer Science, R. Sutton, Ed. Springer US, 1992, vol. 173, pp. 5–32.

[16] Y. Yue and C. Burges, "On using simultaneous perturbation stochastic approximation for ir measures, and the empirical optimality of lambdarank," in *Advances in neural information processing systems*, 2007.

[17] P. Bosman, J. Grahl, and D. Thierens, in *Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5199, pp. 133–143.

[18] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Verlag, 2004.

[19] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process bandits without regret: An experimental design approach," *CoRR*, vol. abs/0912.3995, 2009.

[20] S. Amari and S. Douglas, "Why natural gradient?" in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2, 1998, pp. 1213–1216 vol.2.

[21] http://www.adrl.ethz.ch/software.

[22] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, vol. 15, pp. 1523–1530, 2002.

[23] M. Hutter, C. Remy, M. Hoepflinger, and R. Siegwart, "Scarleth: Design and control of a planar running robot," in *Int. Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 562–567.

[24] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of hyq–a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[25] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, C. D. Remy, and R. Siegwart, "Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion," in *Int. Conf. on Climbing and Walking Robots (CLAWAR)*, 2012.