# Comparing Trust Region Policy Optimization and Natural Policy Gradient in Reinforcement Learning

**Jerke van den Berg** (11034106) **Ruihan Sun** (12516414) **Tom Lieberum** (13253042) **Erik Jenner** (13237896)

## 1 Introduction

Policy optimization algorithms have become very popular in reinforcement learning, as they are applicable in continuous action spaces. It also directly optimizes the thing we really care about, the policy, which has the advantage that for example small parameter updates lead to only small policy changes.

Especially the use of gradient methods for policy optimization has received increasing attention over the last few years (Schulman et al., 2017a; Lillicrap et al., 2019; Schulman et al., 2017b), as they have been proven to be quite useful for challenging sequential decision-making tasks (Lovatto et al., 2019). For example, Proximal Policy Optimization (PPO) has achieved very good performance on both Atari games and continuous control robotic tasks in MuJoCo environments (Schulman et al., 2017b).

Two popular gradient-based policy optimization algorithms are Trust Region Policy Optimization (TRPO) and Natural Policy Gradient (NPG). Both update their parameters in the same direction, but their step sizes are chosen in different ways – specifically, TRPO can be interpreted as adapting its learning rate at each step to guarantee a bound on the KL divergence between old and new policy. In this report, we determine how large these learning rate adaptations are and whether they follow any pattern. We also investigate what effect they have on the KL divergence between old and new policy. This will show whether the additional complexity of TRPO actually helps in terms of the original motivation.

## 2 Background

Gradient descent can be interpreted as the best update which changes the parameters by no more than some constant in terms of their $L^2$ norm (though approximations mean that this does not hold exactly). NPG (Kakade, 2001) and TRPO (Schulman et al., 2017a) instead constrain the KL divergence between old and new policy, arguably a more natural constraint, which also makes the update independent from the chosen parametrization. This optimization problem can be written as

$$\underset{d\theta}{\operatorname{argmax}}\, J(\theta + d\theta) \quad \text{s.t. } \mathrm{D}_{\mathrm{KL}}(\pi_{\theta+d\theta} \| \pi_\theta) < c$$

and it can be shown that it is approximately solved by $d\theta = \alpha F^{-1} \nabla_\theta J$ where $F$ is the Fisher information matrix of the policy and $\alpha$ is a learning rate that depends on $c$ and $\theta$. $J(\theta) = \mathbb{E}_{s\sim\mu_\theta}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[r(s,a)]$ is the expected return under $\pi_\theta$, with the on-policy distribution $\mu_\theta$ of $\pi_\theta$.

For NPG, the learning rate $\alpha$ is chosen empirically and held constant. This means that the KL bound that motivated this method may be violated for two reasons: First, the $\alpha$ that achieves this bound actually depends on $\theta$. And secondly, the derivation of the update rule only makes a quadratic approximation to the KL divergence, which means that there is no guarantee that the KL bound holds, even for the correct $\alpha$.

TRPO addresses these shortcomings by using the adaptive learning rate

$$\alpha = \sqrt{\frac{2c}{(\nabla_\theta J)^T F^{-1} \nabla_\theta J}} \tag{1}$$

that can be approximately derived from the constrained optimization problem[1]. To actually guarantee the bound, it then backtracks, trying smaller learning rates until it finds an update that obeys the KL bound. This learning rate adaptation is done at each step.

However, as investigated by Paul et al. (2019), the advantage of TRPO over NPG diminishes if we

[1] The exact theoretical derivation for TRPO relies on optimizing a lower bound to the objective, which is computationally intractable, so usually the presented method is used in practice.

apply an efficient hyperparameter tuning for NPG. Thus, one could say that TRPO is beneficial mainly in the sense that it relieves the pain of learning rate tuning. But in general, TRPO performes better due to it's adaptive learning rate mechanism since finding the right learning rate for NPG in different environments is demanding.

## 3 Methods

To compare the performance of TRPO and NPG, we run experiments on Pendulum-v0, CartPole-v1 and Acrobot-v1. These environments are relatively easy to solve and well implemented in the OpenAI gym (Brockman et al., 2016), which allows us to study our research question more easily. Since they are also computationally inexpensive, we are able to run a relatively large number of experiments. They cover both discrete action spaces (Acrobot and CartPole) as well as continuous ones (Pendulum). We include the latter because one of the advantages of policy optimization methods is the ease of handling continuous action spaces, so it is particularly interesting how the algorithms perform in such environments.

We train an Actor-Critic model with TRPO and with NPG on all of those environments. For discrete action spaces, we use a categorical distribution to model the policy, for continuous spaces we use a normal distribution with input-dependent variance and mean.

Both actor and critic are represented by a multilayer perceptron with two hidden layers, 64 neurons per hidden layer and ReLU activation functions. In the Pendulum environment, the actor outputs the mean and log-variance of a Gaussian distribution over actions. In the discrete environments, it outputs logits for a categorical distribution.

As an update target we used generalized advantage estimation (GAE) (Schulman et al., 2018). We used the same hyperparameters for GAE across all of them, namely $\gamma = 0.9$ and $\lambda = 0.97$. These values were chosen by performing a restricted grid-search for the TRPO algorithm in the Pendulum-v0 environment, since the choice seemed to have the largest impact on performance in this environment. We ran each parameter combination on ten seeds (0-9). We first varied $\gamma$ for $\lambda = 1$, which is shown in fig. 1 As can be seen the optimal value lies somewhere around $\gamma = 0.9$, which we chose for subsequent experiments. For this $\gamma$, we varied $\lambda$, as can be seen in fig. 2. In contrast to $\gamma$, the precise value of $\lambda$ does not seem to have a very strong influence on performance. We chose a value of 0.97 for subsequent experiment, due to the lower variance compared to $\lambda = 0$. Other values would also have been defensible.[2]

Ideally, we would also have performed a similar search in the other environments but that would have taken significant time investment that would have detracted from other experiments. If the precise performance mattered for us, tuning these parameters on each environment would have been essential. However, the variables we measure are step size and the size of the KL updates, and we only discuss those results qualitatively. We believe that within the range of values that lead to convergence and reasonable results, the precise values of $\gamma$ and $\lambda$ don't have a qualitative influence on those results.

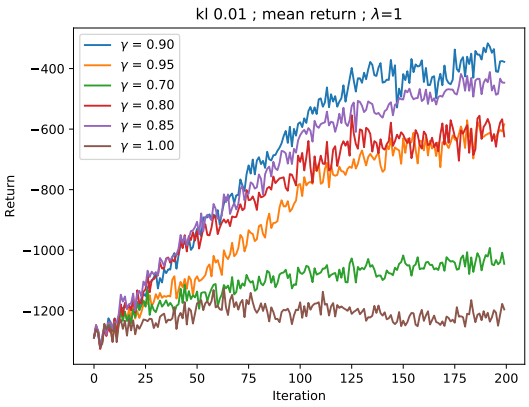



Figure 1: Return plot over iterations for different $\gamma$ values for GAE TRPO on the Pendulum environment

After these preliminary hyperparameter tests, all further experiments are run with five different seeds (10-14) to avoid accidentally fitting to our earlier seeds. We use multiple seeds to determine the range of fluctuations across seeds (Henderson et al. (2018) demonstrates that outcomes can vary strongly across different random seeds).

Our implementation of NPG and TRPO is based on Lee (2019), with some modifications to use the same targets in both cases and read out step size and KL changes.

Our code and data can be found on https://github.com/TomFrederik/rl_reproducibility.

[2]We did not perform an exhaustive search of all combinations for $\gamma$ and $\lambda$, since this would have taken too much compute and we already achieved satisfactory performance.

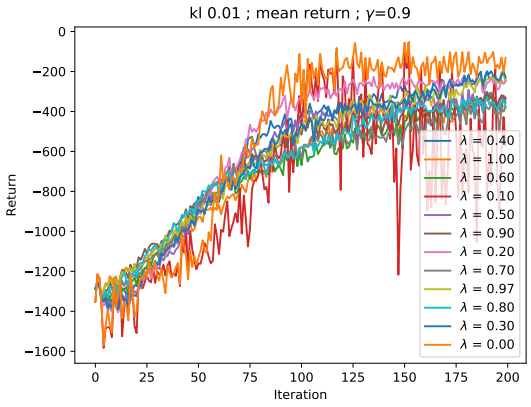


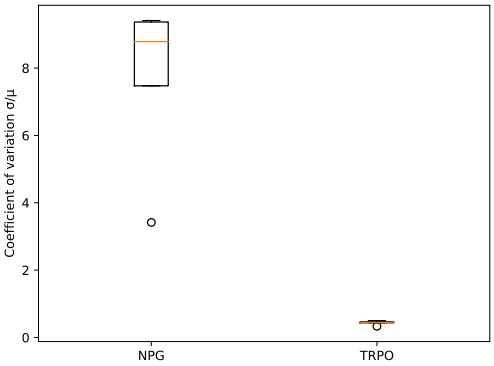

Figure 2: Return plot over iterations for different $\lambda$ values for GAE TRPO on the Pendulum environment

## 4 Experiments and Results

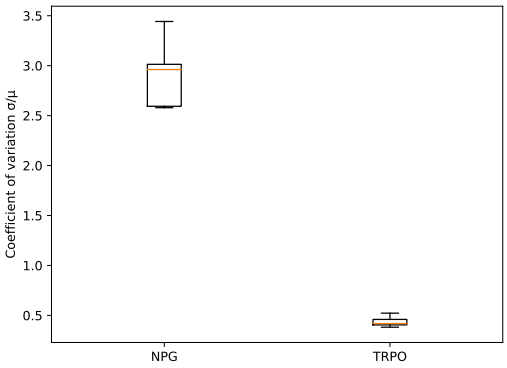
Figure 3 shows the coefficient of variation of the size of KL divergences between old and new policy, i.e. standard deviation divided by mean. Explicitly, we measured the KL divergence of each policy change over 200 iterations and then calculated the standard variation and mean over those 200 values. The coefficient of variation is significantly larger for NPG than for TRPO across all three environments. The underlying KL divergences are all shown in fig. 6 in the appendix.

We can also observe in fig. 6 that NPG sometimes takes very large steps (in terms of KL divergence) and much smaller steps most of the time. For TRPO we can see that it also varies a lot, but only within the bounds set by the maximum KL constraint, so that the coefficient of variation is much smaller. So NPG has a constant step size but this does not translate to constant KL divergences. In contrast, TRPO varies its step size to achieve KL divergences that are closer to constant.

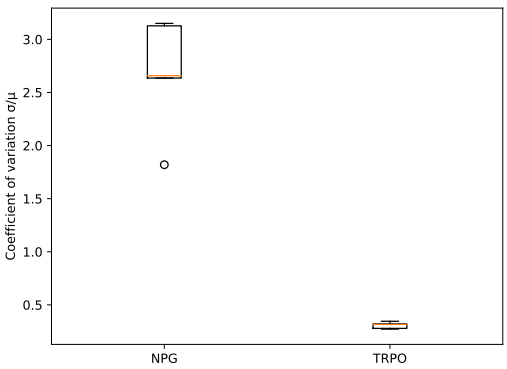
We can also see this from a different perspective, by displaying the step-size. Here we expect a constant line for NPG and a variable step-size for TRPO. The step-sizes for the Pendulum-v0 environment as well as Cartpole and Acrobot are shown in fig. 4. For NPG, we achieved decent performance with a learning rate of 0.1, shown as a constant line in the plots. Comparing the step sizes directly is not meaningful because the two methods require different kinds of parameters (a KL bound for TRPO and a learning rate for NPG). But we can compare the trends in step sizes: for NPG it is constant by design, whereas for TRPO it varies and tends to increase in our experiments.




(a) Pendulum-v0




(b) CartPole-v1




(c) Acrobot-v1

Figure 3: Coefficient of variation (CV) of KL updates for NPG and TRPO. The CV was calculated across all the updates in one episode. The box plot then shows the spread of the CV over 5 different seeds.

We see that the step size is the lowest in Pendulum, followed by Acrobot and is significantly larger in the CartPole environment. There is a slight general upwards trend in the step size as the policy improves.

## 5 Discussion

Figure 3 clearly demonstrates that the KL update of NPG varies more in terms of its coefficient of variation, compared to TRPO. The difference is highly significant and stable across the three environments we tested. This is what we expected; in section 2, we discussed how TRPO makes some changes to NPG that are motivated by adhering more closely to the KL bound, which should lead to more uniform KL updates.

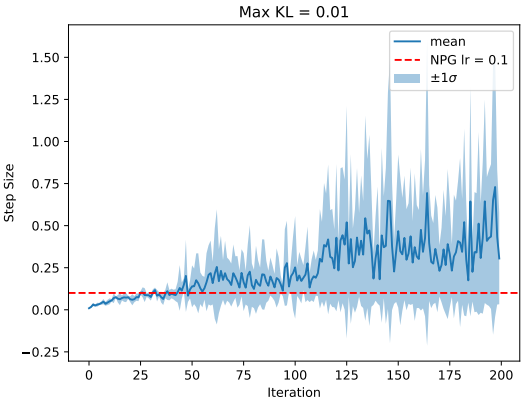
As we can observe from fig. 4, the step-size follows a slight upwards trend, and is thus positively correlated with the return an actor achieves in the environment. One possible hypothesis for this is that as the actor gets better the changes in the objective get smaller as it hits diminishing returns. Thus the gradient in the denominator in eq. (1) becomes smaller, resulting in larger step-sizes.

Furthermore, there is empirical evidence that batch gradient descent with reasonably small batch sizes tends to find flat minima (Keskar et al., 2017). This would mean that the Hessian of the policy, i.e. the Fisher matrix $F$, would tend towards small values which further decreases the step size $\alpha$ from eq. (1). Even before convergence sets in, it is plausible that the stochasticity of mini-batches already steers the parameters towards flat parts of the policy space.

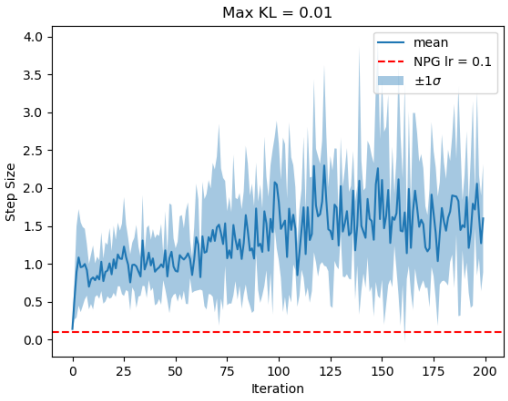
However, these hypotheses are quite speculative, especially given that the upwards trend in step size should be taken with a grain of salt in light of the large variance.

## 6 Conclusion

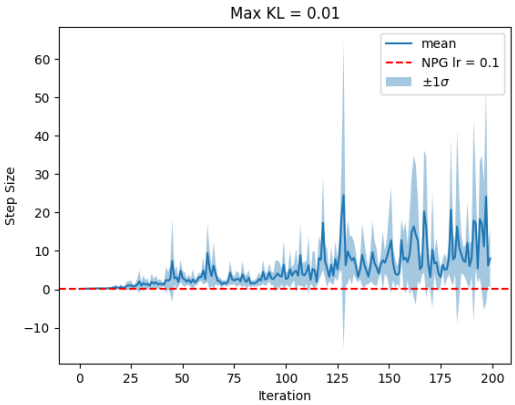
In this report we investigated the adaptability of the step-size in TRPO, and compared it to the NPG algorithm. We have shown that TRPO indeed changes its step size over time and that as a result the variation in the KL divergence between old and new policies is significantly lower for TRPO than for NPG. In addition to this, the step-size of TRPO seems to get bigger as the performance improves, as discussed in section 5.




(a) Pendulum-v0




(b) Acrobot-v1




(c) Cartpole-v1

Figure 4: Step-size for TRPO, averaged over 5 seeds. Kl threshold of 0.01. $\lambda = 0.97, \gamma = 0.9$.

## References


Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym.

Peter Henderson, R. Islam, Philip Bachman, Joelle Pineau, D. Precup, and D. Meger. 2018. Deep reinforcement learning that matters. *ArXiv*, abs/1709.06560.

Sham M. Kakade. 2001. A natural policy gradient. In *NeurIPS*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima.

Woongwon Lee. 2019. Policy gradient algorithms. `https://github.com/reinforcement-learning-kr/pg_travel`.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning.

Ângelo Gregório Lovatto, Thiago Pereira Bueno, and Leliane Nunes Barros. 2019. Analyzing the effect of stochastic transitions in policy gradients in deep reinforcement learning. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 413–418. IEEE.

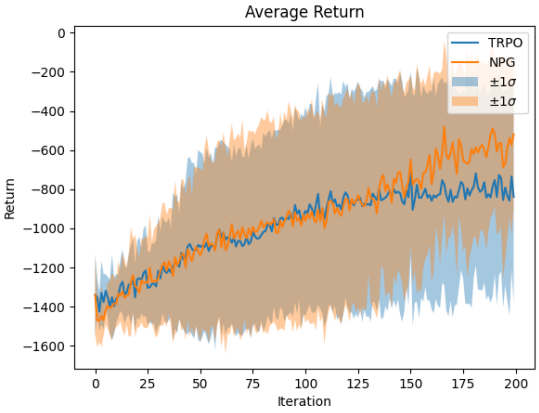
Supratik Paul, Vitaly Kurin, and Shimon Whiteson. 2019. Fast efficient hyperparameter tuning for policy gradient methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4616–4626. Curran Associates, Inc.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization.

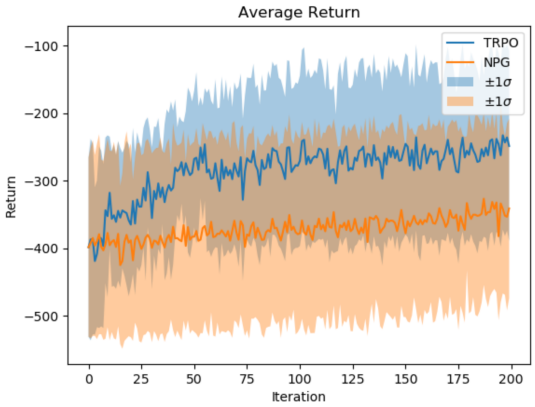
John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-dimensional continuous control using generalized advantage estimation.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

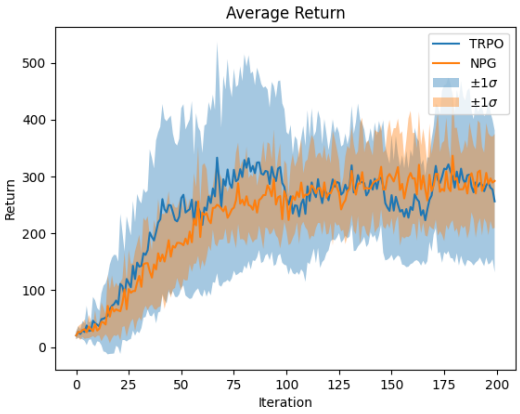
## Appendix




(a) Pendulum-v0

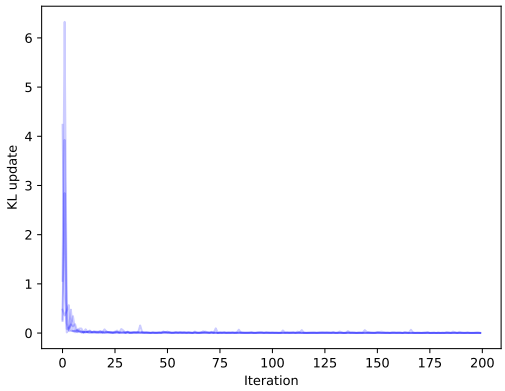



(b) Acrobot-v1

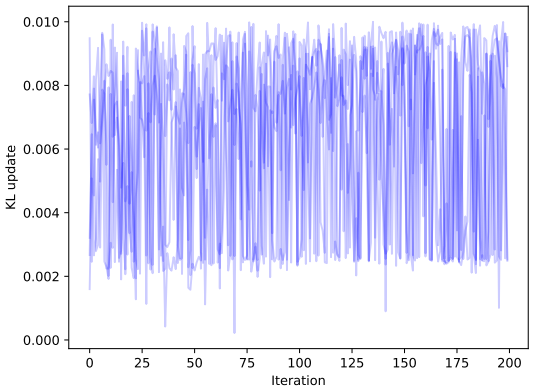



(c) Cartpole-v1

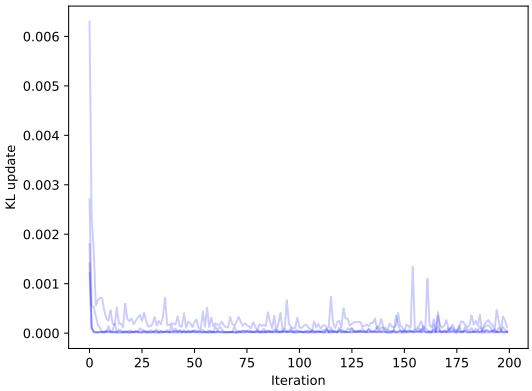
Figure 5: Average returns for NPG and TRPO, averaged over 5 seeds. Max KL constraint of 0.01. $\lambda = 0.97, \gamma = 0.9$.



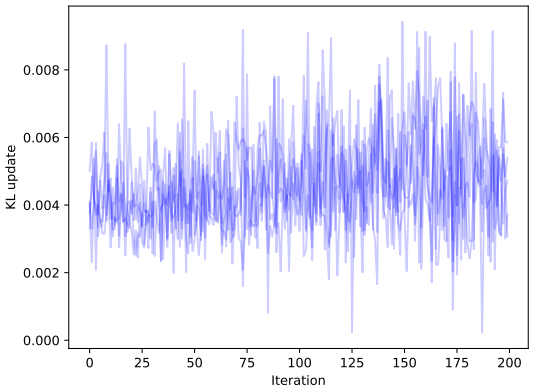

(a) Pendulum-v0 NPG



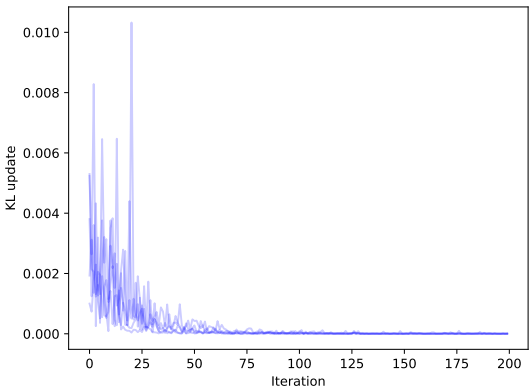

(b) Pendulum-v0 TRPO



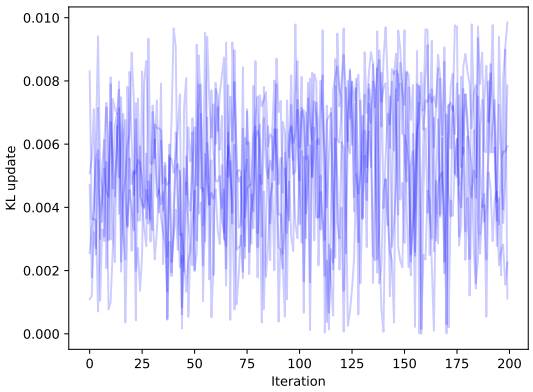

(c) Acrobot-v1 NPG




(d) Acrobot-v1 TRPO




(e) Cartpole-v1 NPG




(f) Cartpole-v1 TRPO

Figure 6: KL divergences for 5 runs with different random seeds (10-14) for the Pendulum, Acrobot and CartPole environments, for NPG (left) and TRPO (right).