Supplementary Materials for: Responsive Safety in Reinforcement Learning by PID Lagrangian Methods

1. Derivations

Here we derive the effects of the proportional and derivative control terms on the Differential Multiplier Method.

1.1. Traditional Lagrangian method

Start from continuous Lagrangian dynamics for objective f and constraint g:

$$\mathcal{L} = f(\mathbf{x}) + \lambda g(\mathbf{x}) \tag{1}$$

$$\dot{x}_i = -\frac{\partial f}{\partial x_i} - \lambda \frac{\partial g}{\partial x_i} \tag{2}$$

$$\dot{\lambda} = \alpha g(\mathbf{x}) \tag{3}$$

with α a scalar. Taking the time-derivative of $\dot{x}_i$:

$$\ddot{x}_i = -\frac{d}{dt}\left(\frac{\partial f}{\partial x_i}\right) - \dot{\lambda}\frac{\partial g}{\partial x_i} - \lambda \frac{d}{dt}\left(\frac{\partial g}{\partial x_i}\right) \tag{4}$$

$$\frac{d}{dt}\left(\frac{\partial f}{\partial x_i}\right) = \sum_j \frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_i}\right)\frac{dx_j}{dt} = \sum_j \frac{\partial^2 f}{\partial x_i \partial x_j}\dot{x}_j \tag{5}$$

$$\frac{d}{dt}\left(\frac{\partial g}{\partial x_i}\right) = \sum_j \frac{\partial^2 g}{\partial x_i \partial x_j}\dot{x}_j \tag{6}$$

and substituting for $\dot{\lambda}$:

$$\ddot{x}_i = -\sum_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \lambda \frac{\partial^2 g}{\partial x_i \partial x_j}\right)\dot{x}_j - \alpha g(\mathbf{x})\frac{\partial g}{\partial x_i} \tag{7}$$

Defining the damping matrix, A:

$$A_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} + \lambda \frac{\partial^2 g}{\partial x_i \partial x_j} \tag{8}$$

yields the oscillator equation for the i-th component:

$$\ddot{x}_i + \sum_j A_{ij}\dot{x}_j + \alpha g(\mathbf{x})\frac{\partial g}{\partial x_i} = 0 \tag{9}$$

Written in vector form:

$$\ddot{\mathbf{x}} + A\dot{\mathbf{x}} + \alpha g(\mathbf{x})\nabla g = 0 \tag{10}$$

$$A = \nabla^2 f + \lambda \nabla^2 g \tag{11}$$

with $\nabla^2 f$ denoting the Hessian of f.

1.2. Proportional-Integral Method

We amend the differential equation for λ as:

$$\dot{\lambda} = \alpha g(\mathbf{x}) + \beta \dot{g}(\mathbf{x}) = \alpha g(\mathbf{x}) + \beta \sum_j \frac{\partial g}{\partial x_j} \dot{x}_j \tag{12}$$

Substitution into $\ddot{x}$ yields:

$$\ddot{x}_i = -\sum_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \lambda \frac{\partial^2 g}{\partial x_i \partial x_j} \right) \dot{x}_j - \alpha g(\mathbf{x}) \frac{\partial g}{\partial x_i} - \beta \left(\sum_j \frac{\partial g}{\partial x_j} \dot{x}_j \right) \frac{\partial g}{\partial x_i} \tag{13}$$

$$= -\sum_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \lambda \frac{\partial^2 g}{\partial x_i \partial x_j} + \beta \frac{\partial g}{\partial x_i} \frac{\partial g}{\partial x_j} \right) \dot{x}_j - \alpha g(\mathbf{x}) \frac{\partial g}{\partial x_i} \tag{14}$$

giving the same dynamics as the traditional Lagrangian method except with damping matrix modified by addition of a positive-semi-definite term:

$$\ddot{\mathbf{x}} + \left(A + \beta \nabla g \nabla^\top g \right) \dot{\mathbf{x}} + \alpha g(\mathbf{x}) \nabla g = 0 \tag{15}$$

1.3. Integral-Derivative Method

We use the following differential equation for λ:

$$\dot{\lambda} = \alpha g(\mathbf{x}) + \gamma \ddot{g}(\mathbf{x}) \tag{16}$$

Expanding the second-derivative:

$$\ddot{g}(x) = \frac{d}{dt} \left(\frac{d}{dt} g(\mathbf{x}) \right) \tag{17}$$

$$= \frac{d}{dt} \left(\sum_j \frac{\partial g}{x_j} \dot{x_j} \right) \tag{18}$$

$$= \sum_j \left(\frac{d}{dt} \left(\frac{\partial g}{x_j} \right) \dot{x}_j + \frac{\partial g}{x_j} \frac{d}{dt} (\dot{x}_j) \right) \tag{19}$$

$$= \sum_j \left(\dot{x}_j \sum_k \frac{\partial^2 g}{\partial x_k \partial x_j} \dot{x}_k + \frac{\partial g}{\partial x_j} \ddot{x}_j \right) \tag{20}$$

Substituting the full expression for $\dot{\lambda}$ into $\ddot{x}_i$ yields:

$$\ddot{x}_i = -\sum_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \lambda \frac{\partial^2 g}{\partial x_i \partial x_j} \right) \dot{x_j} - \alpha g(\mathbf{x}) \frac{\partial g}{\partial x_i} - \gamma \frac{\partial g}{\partial x_i} \left(\sum_j \left(\sum_k \dot{x}_j \frac{\partial^2 g}{\partial x_j \partial x_k} \dot{x}_k + \frac{\partial g}{\partial x_j} \ddot{x}_j \right) \right) \tag{21}$$

where the terms with coefficient γ are due to the derivative update term in λ. Now included are terms of second-order in the velocity and additional mixing terms on the acceleration. It is instructive to consider the resulting vector equation in full:

$$0 = \ddot{\mathbf{x}} + A\dot{\mathbf{x}} + \alpha g(\mathbf{x}) \nabla g + \gamma \nabla g \dot{\mathbf{x}}^\top \nabla^2 g \dot{\mathbf{x}} + \gamma \nabla g \nabla^\top g \ddot{\mathbf{x}} \tag{22}$$

$$= (I + \gamma \nabla g \nabla^\top g) \ddot{\mathbf{x}} + A\dot{\mathbf{x}} + \left(\alpha g(\mathbf{x}) + \gamma \dot{\mathbf{x}}^\top \nabla^2 g \dot{\mathbf{x}} \right) \nabla g \tag{23}$$

The acceleration terms are coupled together by a positive definite matrix (identity plus the outer product of the vector ∇g with itself). Therefore a form without coupling of acceleration can be restored by left-multiplying by the inverse of this matrix, letting $B = (I + \gamma \nabla g \nabla^\top g)$:

$$\ddot{\mathbf{x}} + B^{-1} A \dot{\mathbf{x}} + \left(\alpha g(\mathbf{x}) + \gamma \dot{\mathbf{x}}^\top \nabla^2 g \dot{\mathbf{x}} \right) B^{-1} \nabla g = 0 \tag{24}$$

The effects of thenew terms from the derivative update rule are discussed in the main text.

1.4. Proportional-Integral-Derivative Method

Finally, the full PID-Lagrangian method has dynamics which combines the independent effects of the proportional and derivative methods:

$$\ddot{\mathbf{x}} + B^{-1}\left(A + \beta \nabla g \nabla^\top g\right)\dot{\mathbf{x}} + \left(\alpha g(\mathbf{x}) + \gamma \dot{\mathbf{x}}^\top \nabla^2 g \dot{\mathbf{x}}\right) B^{-1}\nabla g = 0 \tag{25}$$

2. Training Details

All of our experiments began with randomly initialized agents. The reward-value and cost-value estimators shared parameters with the policy. We used Generalized Advantage Estimation for both reward and cost advantages. The control input is updated once per iteration, which in our settings included multiple gradient updates on the policy. Training batches typically included the end of several trajectories, allowing an estimate of the average episodic sum of costs at each iteration.

Table 1. Experiment hyperparameters.

HYPERPARAMETER	VALUE
LEARNING RATE	3×10^{-4}
NN HIDDEN LAYER SIZE	512
NN NONLINEARITY	$tanh$
BATCH DIMENSION, TIME	128
BATCH DIMENSION, ENVS	104
PPO EPOCHS	8
PPO MINIBATCHES	1
PPO RATIO-CLIP	0.1
DISCOUNT, γ	0.99
λ_{GAE}	0.97
K_I	1
K_P, K_D	AS SPECIFIED
COST SCALING	$1/10$
EXPONENTIAL MOVING AVERAGE, K_P	0.95
EXPONENTIAL MOVING AVERAGE, K_D	0.9
DIFFERENCE ITERATES DELAY, K_D	15
OBSERVATION NORMALIZATION	TRUE
$\hat{\beta}$	1 (UNLESS SPECIFIED)
EXPONENTIAL MOVING AVERAGE, $\hat{\beta}$	0.9

3. Additional Learning Curves

3.1. Cost and Reward Curves for Additional Environments

This section contains learning curves from the experiments used to make the figures showing cost figure-of-merit versus Lagrange multiplier learning rate, K_I, in the main text. The main text includes curves from DOGGOGOAL2, the other environments are shown here.

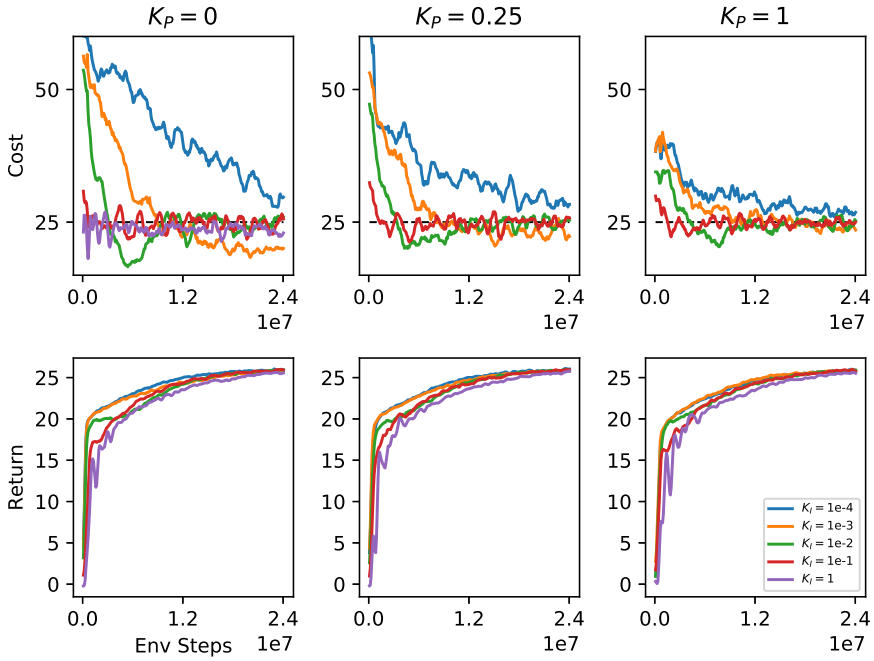

Figure 1. Costs and returns with varying Lagrange multiplier learning rate, K_I, and proportional control coefficient, K_P, in POINTGOAL1, cost-limit=25.

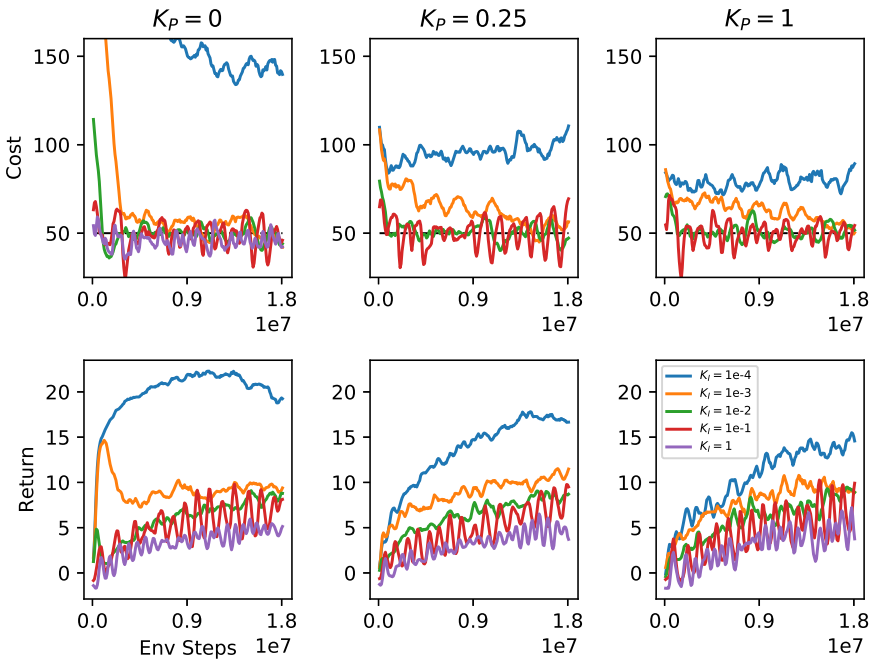

Figure 2. Costs and returns with varying Lagrange multiplier learning rate, K_I, and proportional control coefficient, K_P, in CARBUTTON1, cost-limit=50.

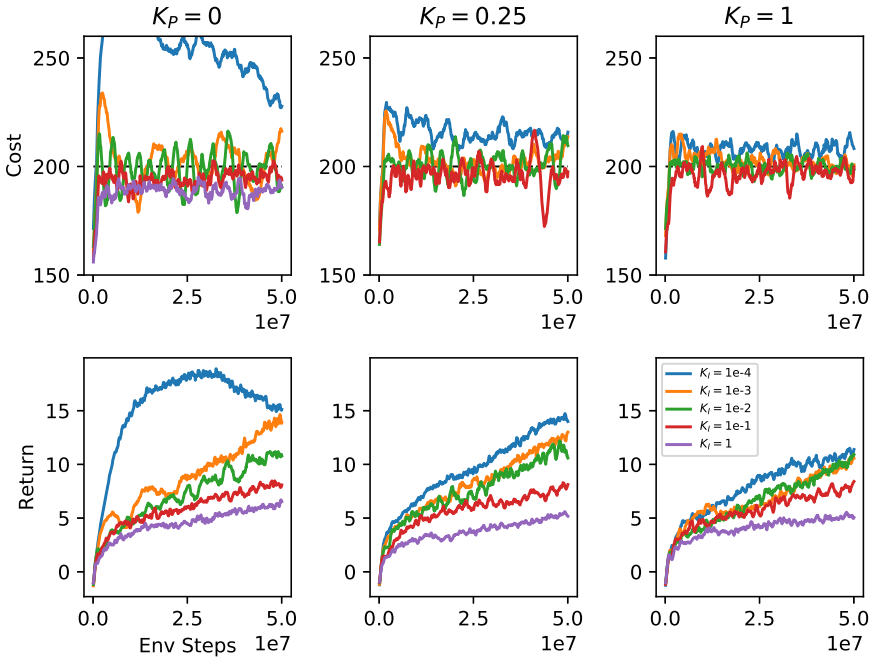

Figure 3. Costs and returns with varying Lagrange multiplier learning rate, K_I, and proportional control coefficient, K_P, in DOGGOBUTTON1, cost-limit=200.

3.2. Derivative-Control Learning Curves

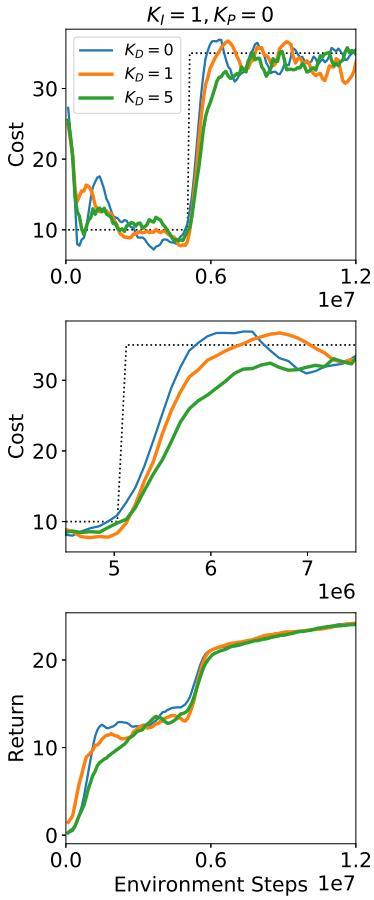

Figure 4. Derivative control slows the increase in cost, causing it to rise more gradually, and reducing overshoot. The cost limit was increased from 10 to 35 at 5M environment steps. Environment: PointGoal1.

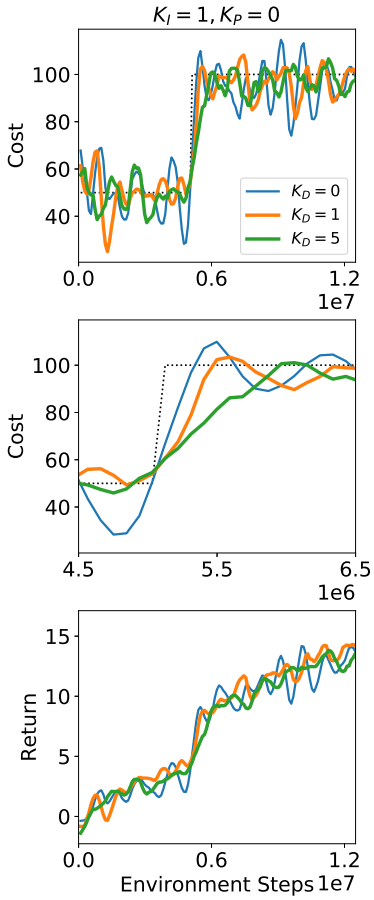

Figure 5. Derivative control slows the increase in cost, causing it to rise more gradually, and reducing overshoot. The cost limit was increased from 50 to 100 at 5M environment steps. Environment: CarButton1.

3.3. Comparison to Unconstrained Algorithms

This section shows learning curves for the same four environments referenced in the main text. These figures demonstrate that the unconstrained algorithm (PPO) does not satisfy the cost constraints, and as a result it achieves higher rewards.

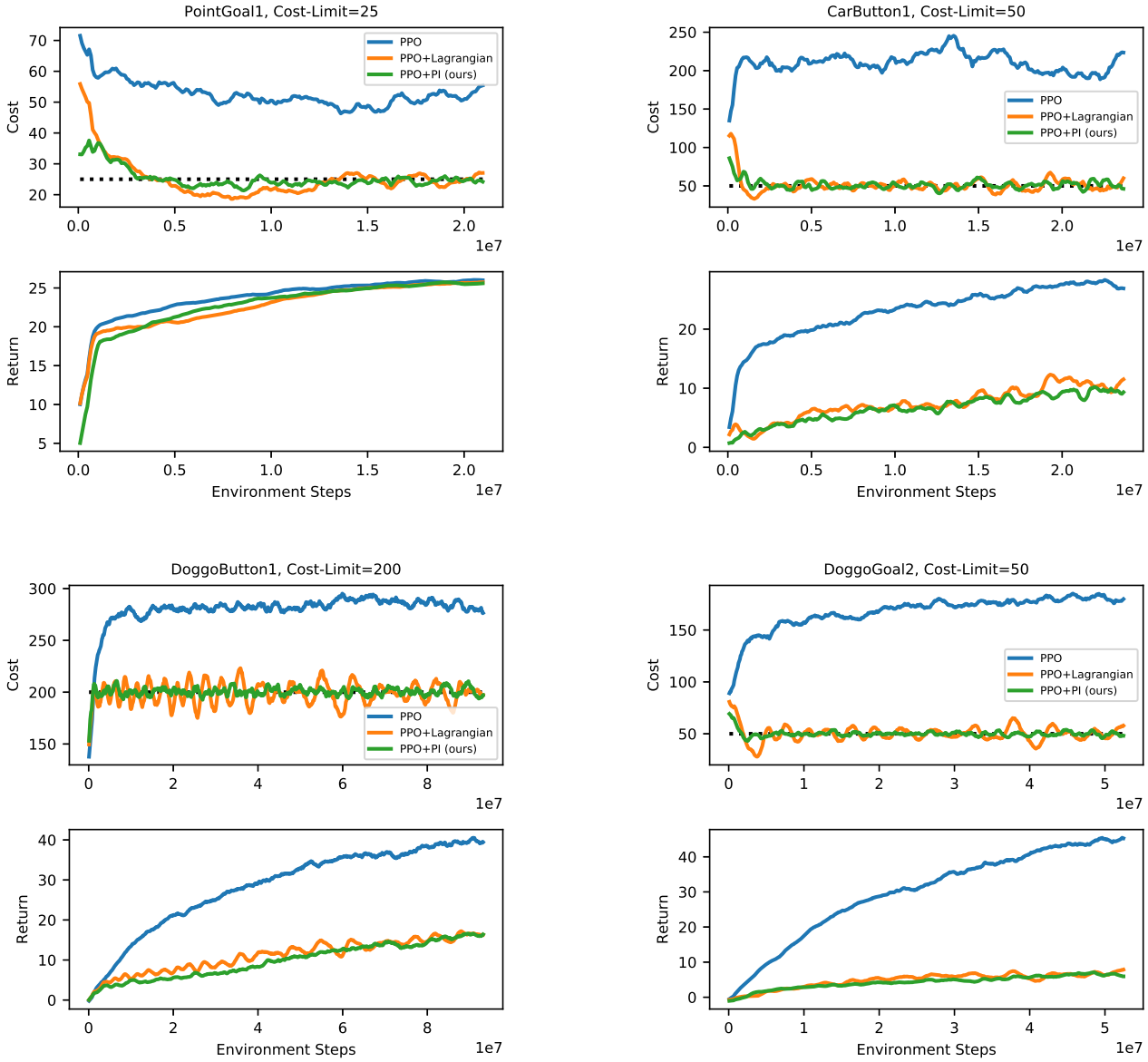

Figure 6. Cost and reward curves for three variants of PPO: unconstrained , Lagrangian, and PI-Controlled. The unconstrained algorithm wildly violates all cost-limits used in our experiments. PPO+Lagrangian and PPO+PI use the same Lagrange multiplier learning rate, K_I.

4. Adaptive Objective-Balancing

Alternative, KL-Based Estimator The magnitudes of the gradients in θ-space might not fully reflect the relative impacts of reward- and cost-learning on agent behavior. Reinforcement learning offers an alternative grounding in policy-space, for example using:

$$\beta_{KL} = \frac{D_{KL}(\pi_{\theta_k} || \pi^R_{\theta_k+1})}{D_{KL}(\pi_{\theta_k} || \pi^C_{\theta_k+1})} \tag{26}$$

Here, $\pi^R_{\theta_{k+1}}$ and $\pi^C_{\theta_{k+1}}$ are hypothetical new policies found using only the reward-objective or only the (un-scaled) cost-objective, respectively. We experimented with this estimator and found it to work, although not as well as the gradient-norm estimator in our cases. Some results are included in the figures below.

Figures We include figures for POINTGOAL1 using I-control and PI-control, and the more challenging DOGGOGOAL2 using I-control. Observe in the un-balanced case ($\hat{\beta} = 1$) that as the controller settings scale with the reward, the learning dynamics remained the same. For example, see $(K_I = 0.1, \rho = 10), (K_I = 0.01, \rho = 1),$ and $(K_I = 0.001, \rho = 0.1)$. Using gradient-based objective balancing (β_∇), however, the learning dynamics were roughly the same across all reward scales, for given controller settings. Alternatively, KL-based objective balancing (β_{KL}) was also effective, but did not produce dynamics as uniform as the gradient-based method.

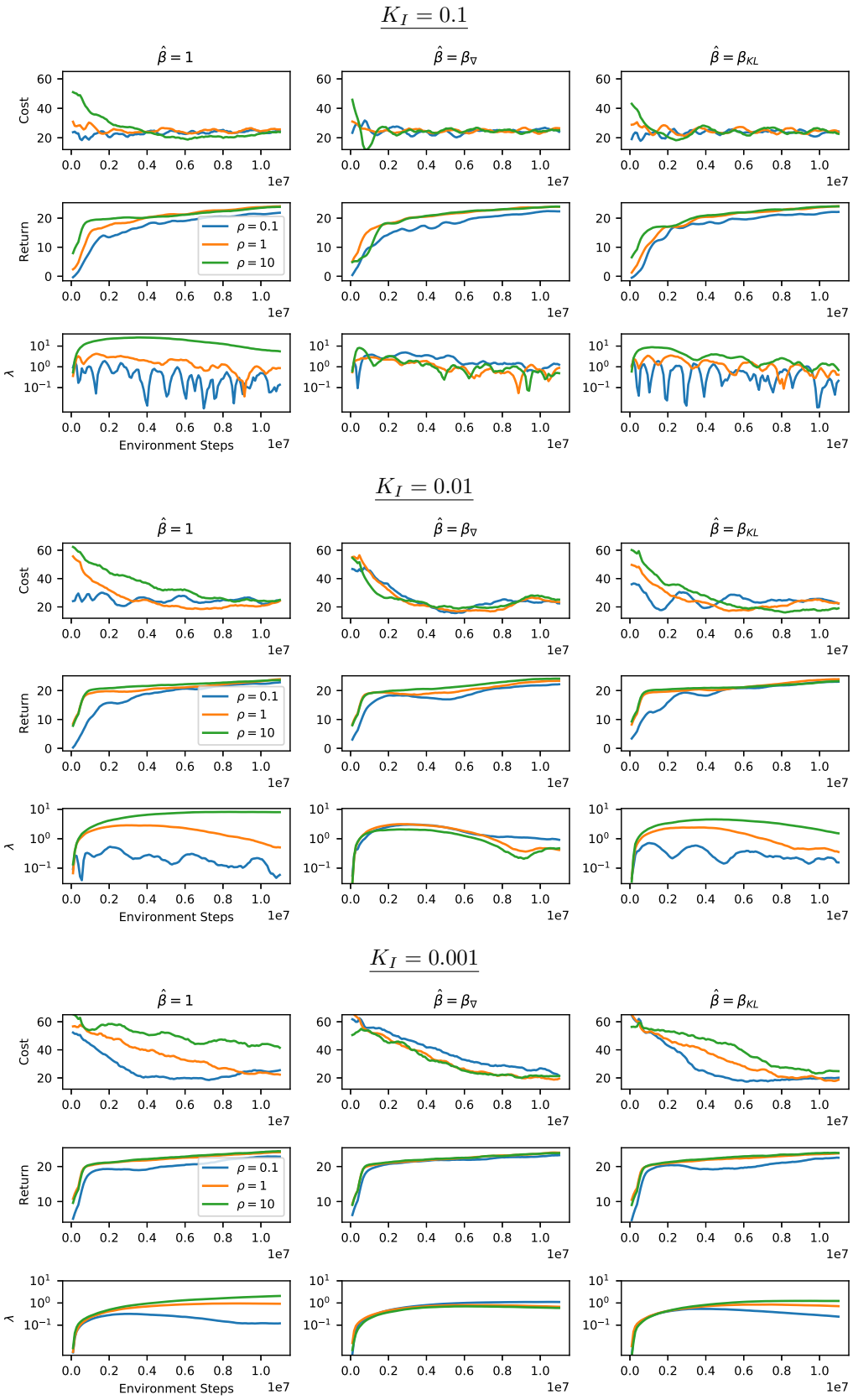

Figure 7. Reward scaling, I-control, POINTGOAL1, cost-limit=25.

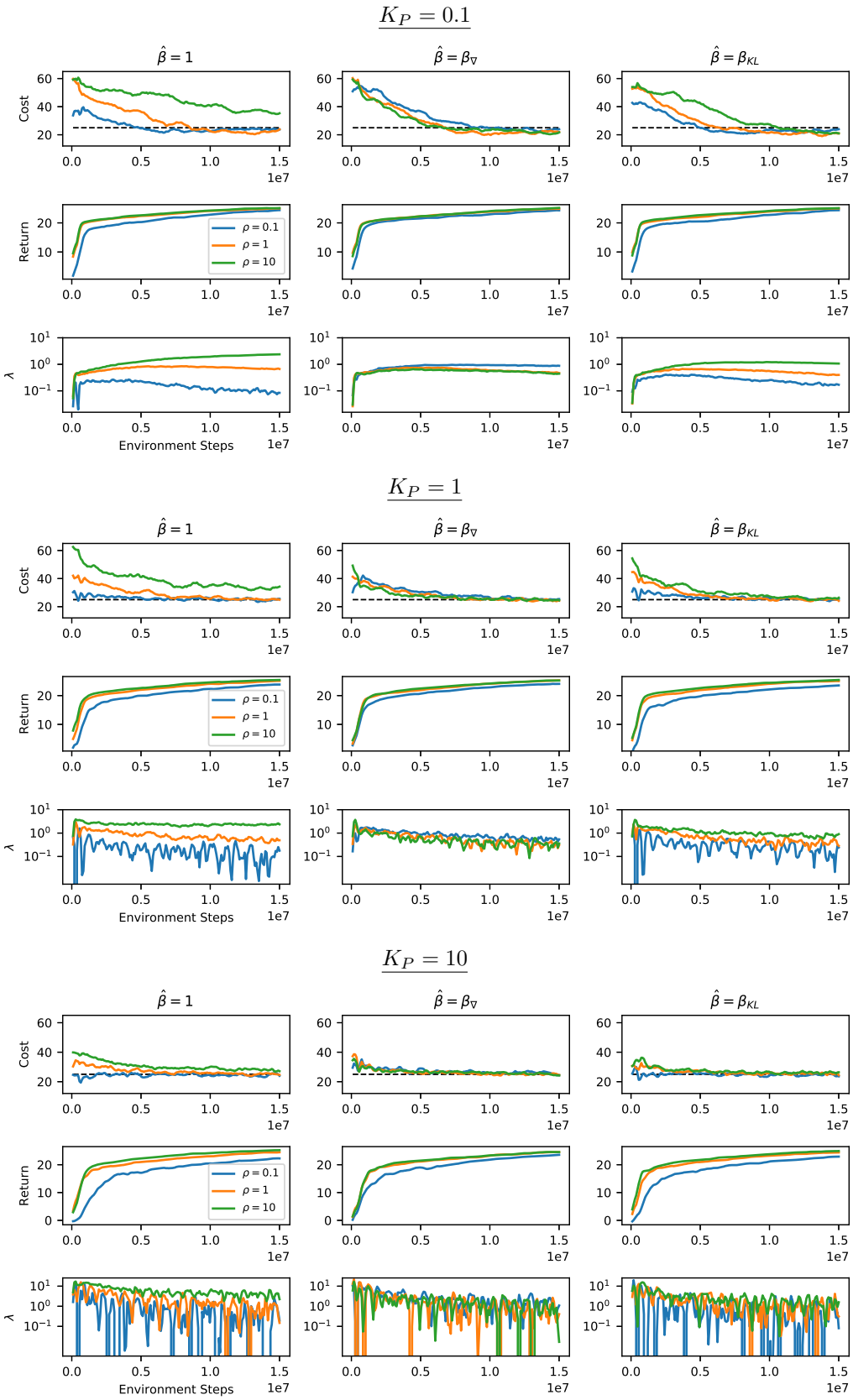

Figure 8. Reward scaling, PI-control with $K_I = 0.001$, POINTGOAL1, cost-limit=25.

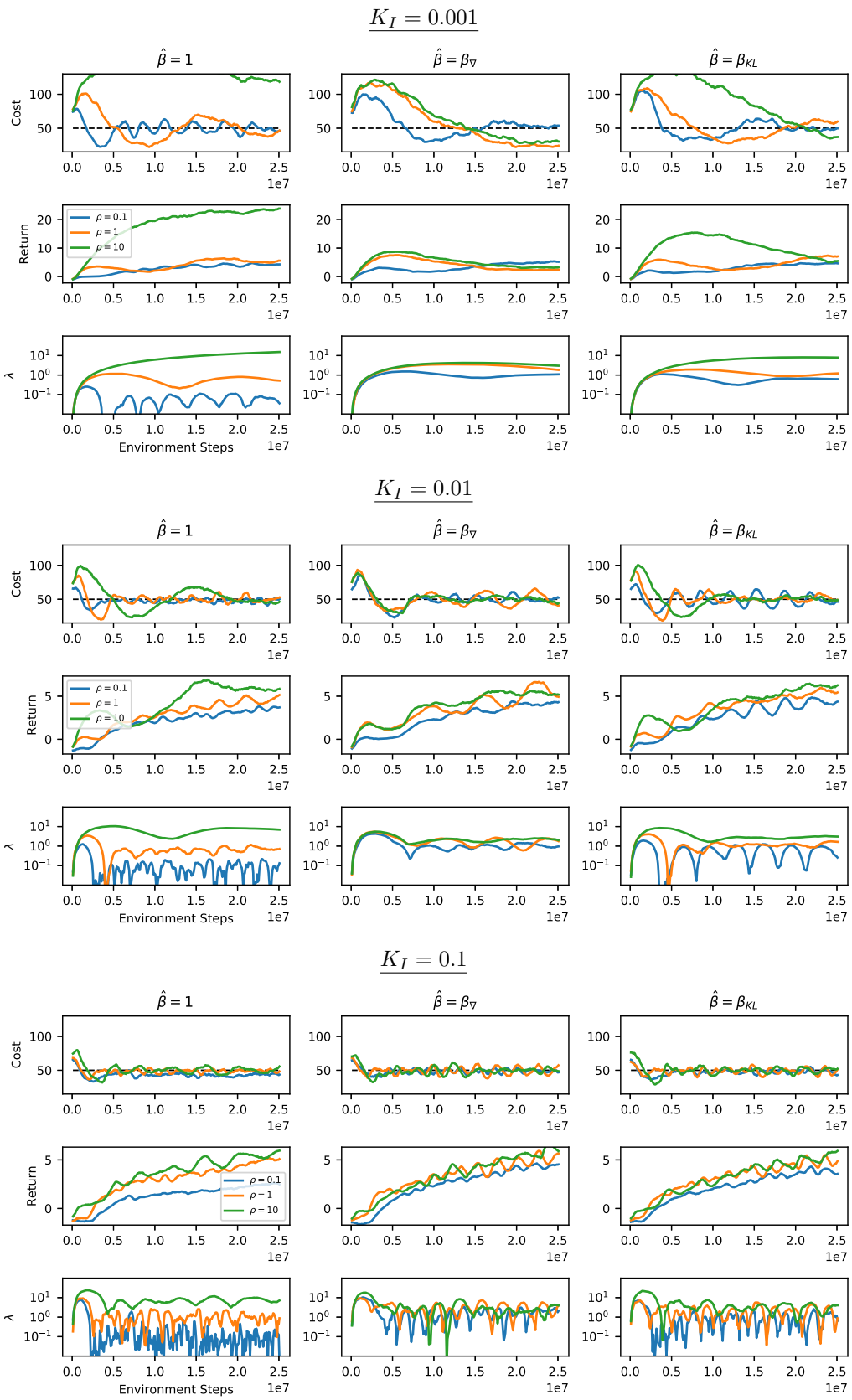

Figure 9. Reward scaling, I-control, DOGGOGOAL2, cost-limit=50.