Supplementary Material for *Conformal Symplectic Optimization for Stable Reinforcement Learning*

Yao Lyu, Xiangteng Zhang, Shengbo Eben Li*, Jingliang Duan, Letian Tao, Qing Xu, Lei He, Keqiang Li

S.I Basis of conformal symplectic optimization

This section introduces some basis of conformal symplectic optimization, including symplectic integrators and splitting methods.

A. Symplectic integrators

A one-step numerical method is called symplectic if the one-step map:

$$z_1 = \phi_h(z_0)$$

is symplectic whenever the method is applied to a smooth, conservative Hamiltonian system. The following theorems show two typical symplectic integrators with respect to the conservative Hamiltonian system $\dot{z} = S\nabla H(z) = \mathcal{C}(z)$ [30].

Theorem 2. *The so-called symplectic Euler methods:*

$$p_{k+1} = p_k - h\nabla_q H(q_k, p_{k+1}), \quad q_{k+1} = q_k + h\nabla_p H(q_k, p_{k+1}), \tag{17}$$

or

$$p_{k+1} = p_k - h\nabla_q H(q_{k+1}, p_k), \quad q_{k+1} = q_k + h\nabla_p H(q_{k+1}, p_k), \tag{18}$$

are symplectic methods of order 1.

Theorem 3. *The Verlet schemes, i.e., leapfrog methods:*

$$\begin{aligned} p_{k+\frac{1}{2}} &= p_k - \frac{h}{2}\nabla_q H\left(q_k, p_{k+\frac{1}{2}}\right), \\ q_{k+1} &= q_k + \frac{h}{2}\left(\nabla_p H\left(q_k, p_{k+\frac{1}{2}}\right) + \nabla_p H\left(q_{k+1}, p_{k+\frac{1}{2}}\right)\right), \\ p_{k+1} &= p_{k+\frac{1}{2}} - \frac{h}{2}\nabla_q H\left(q_{k+1}, p_{k+\frac{1}{2}}\right), \end{aligned} \tag{19}$$

or

$$\begin{aligned} q_{k+\frac{1}{2}} &= q_k + \frac{h}{2}\nabla_p H\left(q_{k+\frac{1}{2}}, p_k\right), \\ p_{k+1} &= p_k - \frac{h}{2}\left(\nabla_q H\left(q_{k+\frac{1}{2}}, p_k\right) + \nabla_q H\left(q_{k+\frac{1}{2}}, p_{k+1}\right)\right), \\ q_{k+1} &= q_{k+\frac{1}{2}} + \frac{h}{2}\nabla_p H\left(q_{k+\frac{1}{2}}, p_{k+1}\right), \end{aligned} \tag{20}$$

are symplectic methods of order 2.

B. Splitting methods

Definition 1 (Splitting methods)**.** *Consider an arbitrary system $\dot{z} = \zeta(z)$ in $\mathbb{R}^{2n}$, and suppose that the vector field is "split" as*

$$\dot{z} = \zeta^{[1]}(z) + \zeta^{[2]}(z).$$

If the exact flow $\varphi_t^{[1]}$ and $\varphi_t^{[2]}$ of the system $\dot{z} = \zeta^{[1]}(z)$ and $\dot{z} = \zeta^{[2]}(z)$ can be calculated explicitly, we can compose their numerical maps $\phi_h^{[1]}$ and $\phi_h^{[2]}$ to get the numerical approximation $\phi_h = \phi_h^{[1]} \circ \phi_h^{[2]}$ of the system $\dot{z} = \zeta(z)$.

Moreover, it is proved that splitting methods in the composition $\phi_h^{[1]} \circ \phi_h^{[2]}$ produces a first-order integrator, while outputs a second-order integrator in the composition $\phi_{h/2}^{[2]} \circ \phi_h^{[1]} \circ \phi_{h/2}^{[2]}$ [30].

C. Derivative of the DLPF algorithm

This section provides a detailed analysis of the derivative of the DLPF algorithm through the discretization of conformal Hamiltonian systems comprising numerous independent microscopic one-dimensional particles. If we choose the leapfrog method (20) to integrate the conservative flow $\varphi_t^{\mathcal{C}}$ and consider the composition $\phi_h = \phi_{h/2}^{\mathcal{D}} \circ \phi_h^{\mathcal{C}} \circ \phi_{h/2}^{\mathcal{D}}$, we obtain

$$\begin{aligned}
q_{k+\frac{1}{2}} &= q_k + \frac{h}{2}\nabla_p H\left(q_{k+\frac{1}{2}}, e^{-\frac{1}{2}rh}p_k\right), \\
p_{k+\frac{1}{2}} &= e^{-\frac{1}{2}rh}p_k - \frac{h}{2}\left(\nabla_q H\left(q_{k+\frac{1}{2}}, e^{-\frac{1}{2}rh}p_k\right) + \nabla_q H\left(q_{k+\frac{1}{2}}, p_{k+\frac{1}{2}}\right)\right), \\
q_{k+1} &= q_{k+\frac{1}{2}} + \frac{h}{2}\nabla_p H\left(q_{k+\frac{1}{2}}, p_{k+\frac{1}{2}}\right), \\
p_{k+1} &= e^{-\frac{1}{2}rh}p_{k+\frac{1}{2}}.
\end{aligned} \tag{21}$$

Then, let us consider the classical Hamiltonian with each one-dimensional particle possessing the same mass m, i.e., $H(\theta, p) = \sum_{i=1}^{n} \frac{p_i^2}{2m} + J(\theta)$, and replace it into (21). Introducing the change of variables (7), we have

$$\begin{aligned}
\theta_{k+\frac{1}{2}} &= \theta_k - \frac{1}{2}\alpha\sqrt{\beta_1}v_k, \\
v_{k+\frac{1}{2}} &= \sqrt{\beta_1}v_k + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+1} &= \theta_{k+\frac{1}{2}} - \frac{1}{2}\alpha v_{k+\frac{1}{2}}, \\
v_{k+1} &= \sqrt{\beta_1}v_{k+\frac{1}{2}}.
\end{aligned} \tag{22}$$

Note that the algorithm remains the same if we replace successive updating rules. Thus, we rewrite them into two-step updating rules and obtain

$$\begin{aligned}
v_{k+\frac{1}{2}} &= \sqrt{\beta_1} \cdot \sqrt{\beta_1}v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right) \\
&= \beta_1 v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+\frac{3}{2}} &= \theta_{k+1} - \frac{1}{2}\alpha\sqrt{\beta_1}v_{k+1} \\
&= \theta_{k+\frac{1}{2}} - \frac{1}{2}\alpha v_{k+\frac{1}{2}} - \frac{1}{2}\alpha\sqrt{\beta_1} \cdot \sqrt{\beta_1}v_{k+\frac{1}{2}} \\
&= \theta_{k+\frac{1}{2}} - \frac{1}{2}\alpha(\beta_1+1)v_{k+\frac{1}{2}}.
\end{aligned} \tag{23}$$

The DLPF algorithm is a second-order conformal symplectic integrator [26], whose pseudocode is shown in Algorithm 5. It is worth noting that θ_k and v_k in Algorithm 5 actually means $\theta_{k+\frac{1}{2}}$ and $v_{k-\frac{1}{2}}$ in (23), respectively. But the only distinction is that their subscripts are different, which is written on purpose for the convenience and unification of the pseudocode.

Algorithm 5 Dissipative leapfrog (DLPF) algorithm [22]

```
Input: parameters of neural network $\theta_0$ and their conjugate momenta $v_0$, learning rate $\alpha > 0$, first-order momentum coefficient $0 < \beta_1 < 1$
1: for $k = 0$ to $N - 1$ do
2:     $v_{k+1} = \beta_1 v_k + (1-\beta_1)\nabla J(\theta_k)$
3:     $g_k = \frac{1}{2}(\beta_1 + 1)v_{k+1}$
4:     $\theta_{k+1} = \theta_k - \alpha g_k$
5: end for
```

S.II More details on RAD

A. Relativistic Hamiltonian

Here, we derive the relativistic Hamiltonian in the form of (11). In special relativity, the well-known mass-energy equation is

$$E = m\gamma_r c^2,$$

where E is the total energy of a single particle, m is the rest mass of the single particle, $\gamma_r = \frac{1}{\sqrt{1-s^2/c^2}}$ is the relativistic coefficient, s is the speed of the single particle, and c is the speed of light. Therefore, the rest energy, in which case the $s = 0$, is

$$E_0 = mc^2,$$

and the momentum of a single particle is

$$p_i = m\gamma_r s.$$

Here, it is easy to obtain the following equation:

$$c^2 p_i^2 + E_0^2 = \frac{m^2 c^4}{1 - s^2/c^2} = E^2,$$

Hence, the kinetic energy of the single particle is

$$T(p_i) = E - E_0 = \sqrt{c^2 p_i^2 + E_0^2} - E_0 = c\sqrt{p_i^2 + m^2 c^2} - E_0.$$

Since the rest energy E_0 is constant, having no contribution to the system's canonical equations, we can ignore it and have the Hamiltonian

$$H(q,p) = T(p) + U(q) = \sum_i T(p_i) + U(q) = \sum_i c\sqrt{p_i^2 + m^2 c^2} + U(q).$$

Finally, we obtain (11) by replacing the potential energy $U(q)$ with the objective $J(\theta)$.

B. Derivative of the second-order RAD

Consider the relativistic Hamiltonian (11) and replace it into (21), we receive the following integrator:

$$\begin{aligned}
\theta_{k+\frac{1}{2}} &= \theta_k + \frac{hc}{2}\frac{e^{-\frac{1}{2}rh}}{\sqrt{e^{-rh}p_k^2 + m^2c^2}}p_k, \\
p_{k+\frac{1}{2}} &= e^{-\frac{1}{2}rh}p_k - h\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+1} &= \theta_{k+\frac{1}{2}} + \frac{hc}{2}\frac{1}{\sqrt{p_{k+\frac{1}{2}}^2 + m^2c^2}}p_{k+\frac{1}{2}}, \\
p_{k+1} &= e^{-\frac{1}{2}rh}p_{k+\frac{1}{2}}.
\end{aligned}$$

Introducing the changes of variables (7), thus we obtain

$$\begin{aligned}
\theta_{k+\frac{1}{2}} &= \theta_k - \frac{\alpha/2}{\sqrt{\delta^2 v_k^2 + \frac{1}{\beta_1}}}v_k, \\
v_{k+\frac{1}{2}} &= \sqrt{\beta_1}v_k + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+1} &= \theta_{k+\frac{1}{2}} - \frac{\alpha/2}{\sqrt{\delta^2 v_{k+\frac{1}{2}}^2 + 1}}v_{k+\frac{1}{2}}, \\
v_{k+1} &= \sqrt{\beta_1}v_{k+\frac{1}{2}}.
\end{aligned}$$

Since replacing successive updating rules does not change the algorithm, rewriting them into two-step updating rules as

$$\begin{aligned}
v_{k+\frac{1}{2}} &= \sqrt{\beta_1}\cdot\sqrt{\beta_1}v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right)\\
&= \beta_1 v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right),\\
\theta_{k+\frac{3}{2}} &= \theta_{k+1} - \frac{\alpha/2}{\sqrt{\delta^2 v_{k+1}^2 + \frac{1}{\beta_1}}}v_{k+1}\\
&= \theta_{k+\frac{1}{2}} - \frac{\alpha/2}{\sqrt{\delta^2 v_{k+\frac{1}{2}}^2 + 1}}v_{k+\frac{1}{2}} - \frac{\alpha/2}{\sqrt{\delta^2\beta_1 v_{k+\frac{1}{2}}^2 + \frac{1}{\beta_1}}}\sqrt{\beta_1}v_{k+\frac{1}{2}}\\
&= \theta_{k+\frac{1}{2}} - \left(\frac{1}{\sqrt{\delta_{k+\frac{1}{2}}^2 + 1}} + \frac{1}{\sqrt{\delta^2 v_{k+\frac{1}{2}}^2 + \frac{1}{\beta_1^2}}}\right)\frac{\alpha}{2}v_{k+\frac{1}{2}},
\end{aligned}$$

and we immediately receive the original form of the second-order RAD related to the relativistic Hamiltonian (11).

S.III Basis of other optimization algorithms

This section introduces some basis of the other integrators involved in this paper, including SGD, NAG and ADAM.

A. Stochastic gradient descent algorithm

SGD is precisely a forward Euler discretization for the differential equation $\dot{\theta} = -\nabla J(\theta)$ [14], and its pseudocode is shown in Algorithm 6.

Algorithm 6 Stochastic gradient descent (SGD) algorithm [14]

```
Input: parameters of neural network $\theta_0$, learning rate $\alpha > 0$
1: for $k = 0$ to $N-1$ do
2:     $g_k = \nabla J(\theta_k)$
3:     $\theta_{k+1} = \theta_k - \alpha g_k$
4: end for
```

B. Nesterov accelerated gradient algorithm

It is worth noting that there is a close similarity between DLPF and NAG [16]. DLPF would be exactly NAG if we replace $\theta_{k+\frac{1}{2}}$ with θ_k in the third equation of (22) as

$$\begin{aligned}
\theta_{k+\frac{1}{2}} &= \theta_k - \frac{1}{2}\alpha\sqrt{\beta_1}v_k, \\
v_{k+\frac{1}{2}} &= \sqrt{\beta_1}v_k + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+1} &= \theta_k - \frac{1}{2}\alpha v_{k+\frac{1}{2}}, \\
v_{k+1} &= \sqrt{\beta_1}v_{k+\frac{1}{2}}.
\end{aligned}$$

Rewriting them into two-step updating rules like DLPF, we have

$$\begin{aligned}
v_{k+\frac{1}{2}} &= \sqrt{\beta_1}\cdot\sqrt{\beta_1}v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right) \\
&= \beta_1 v_{k-\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right), \\
\theta_{k+\frac{3}{2}} &= \theta_{k+1} - \frac{1}{2}\alpha\sqrt{\beta_1}v_{k+1} \\
&= \theta_k - \frac{1}{2}\alpha v_{k+\frac{1}{2}} - \frac{1}{2}\alpha\sqrt{\beta_1}\cdot\sqrt{\beta_1}v_{k+\frac{1}{2}} \\
&= \theta_{k+\frac{1}{2}} + \frac{1}{2}\alpha\sqrt{\beta_1}v_k - \frac{1}{2}\alpha v_{k+\frac{1}{2}} - \frac{1}{2}\alpha\beta_1 v_{k+\frac{1}{2}} \\
&= \theta_{k+\frac{1}{2}} - \frac{1}{2}\alpha(1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right) - \frac{1}{2}\alpha\beta_1 v_{k+\frac{1}{2}} \\
&= \theta_{k+\frac{1}{2}} - \frac{1}{2}\alpha\left(\beta_1 v_{k+\frac{1}{2}} + (1-\beta_1)\nabla J\left(\theta_{k+\frac{1}{2}}\right)\right).
\end{aligned}$$

This is the famous NAG algorithm, whose pseudocode is shown in Algorithm 7. Note that we adjust the subscripts for the convenience and unification of the pseudocode as the same in Algorithm 5. Intuitively, pulling the updating base backwards, i.e., replacing $\theta_{k+\frac{1}{2}}$ with θ_k in the third updating rule of (22), introduces unreal dissipation into the original dynamical system [26]. Therefore, although NAG is a first-order integrator of the classical Hamiltonian system, it is not conformal symplectic. Indeed, while DLPF exactly preserves the same dissipation of the continuous-time system, NAG introduces some extra contraction or expansion to the symplectic form, thus changing the behavior of the original system slightly.

Algorithm 7 Nesterov accelerated gradient (NAG) algorithm [16]

```
Input: parameters of neural network $\theta_0$, learning rate $\alpha > 0$
1: for $k = 0$ to $N-1$ do
2:     $v_{k+1} = \beta_1 v_k + (1-\beta_1)\nabla J(\theta_k)$
3:     $g_k = \frac{1}{2}(\beta_1 v_{k+1} + (1-\beta_1)\nabla J(\theta_k))$
4:     $\theta_{k+1} = \theta_k - \alpha g_k$
5: end for
```

C. Adaptive moment gradient algorithm

ADAM is hailed as the most promising algorithm for stochastic nonconvex optimization [31]. It has been utilized in many DL problems and has proven experimentally effective. This algorithm estimates the gradients' first-order and secondary raw moments, then individually adjusts the effective learning rates of different parameters. Its pseudocode from the original paper is shown in Algorithm 8, wherein every operation on vector is element-wise [18].

Algorithm 8 Adaptive moment estimation-original (ADAM) algorithm (original) [31]

```
Input: parameters of neural network $\theta_0$, first-order momenta $v_0$, second-order momenta $y_0$, learning rate $\alpha > 0$, first-order momentum coefficient $0 < \beta_1 < 1$, second-order momentum coefficient $0 < \beta_2 < 1$, rational factor $\hat{\epsilon} > 0$
1: for $k = 0$ to $N - 1$ do
2:     $v_{k+1} = \beta_1 v_k + (1 - \beta_1) \nabla J(\theta_k)$ (Update biased first-order moment estimate)
3:     $y_{k+1} = \beta_2 y_k + (1 - \beta_2) (\nabla J(\theta_k))^2$ (Update biased secondary raw moment estimate)
4:     $\hat{v}_{k+1} = v_{k+1} / \left(1 - \beta_1^{k+1}\right)$ (Compute bias-corrected first-order moment estimate)
5:     $\hat{y}_{k+1} = y_{k+1} / \left(1 - \beta_2^{k+1}\right)$ (Compute bias-corrected secondary raw moment estimate)
6:     $g_k = \hat{v}_{k+1}$ (Estimate gradients of the objective function)
7:     $\alpha_k = \frac{\alpha}{\sqrt{\hat{y}_{k+1}} + \hat{\epsilon}}$ (Adjust effective learning rates)
8:     $\theta_{k+1} = \theta_k - \alpha_k g_k$
9: end for
```

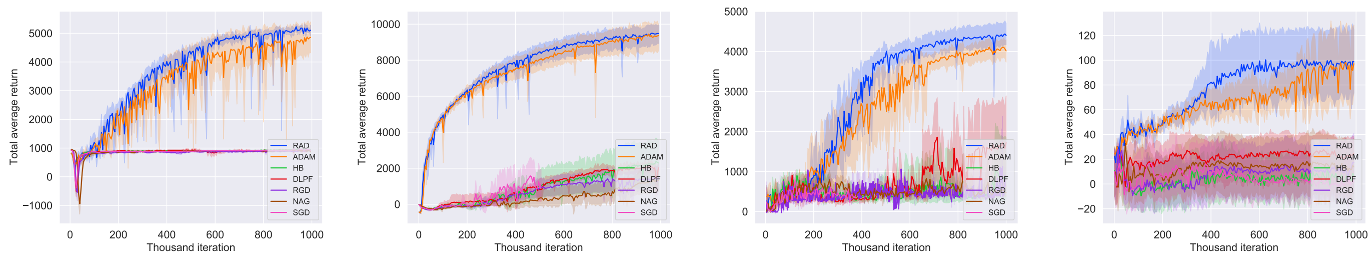
We can compactly write the updating rules by integrating the bias-correction steps into their following steps, thus formulating ADAM in the same format as other algorithms shown in this paper (see Algorithm 4). Note that the rational factor ϵ prevents the denominator of the effective learning rate α_k from zero. ADAM has two essential properties, including its moment estimation of gradients and naturally performing self-adaption on learning rates.

S.V Additional experiments

This section presents additional experimental results not included in the main body due to page limitations.

A. *MuJoCo tests on TD3*

In addition to using SAC, we also validate RAD on various MoJoCo tasks employing the TD3 algorithm. The results demonstrate consistent superiority of RAD across all tasks (see Fig. 8), with a notable 9.7% improvement over ADAM specifically in the Walker2d-v3 task (see Table III). These findings highlight the efficacy and versatility of RAD when applied to different RL algorithms.



(a) Ant-v3 (b) HalfCheetah-v3 (c) Walker2d-v3 (d) Swimmer-v3

Fig. 8: Policy performance of TD3 on MuJoCo tasks. The solid lines correspond to the mean, and the shaded regions correspond to the 95% confidence interval over five runs.

TABLE III: Policy performance of TD3 on MuJoCo tasks over five runs. The maximum value for each task is bolded. $\pm$ corresponds to the standard deviation. DIV represents early divergence. $\Uparrow$ denotes the relative improvement of RAD compared to ADAM.

Tasks	Ant-v3	HalfCheetah-v3	Walker2d-v3	Swimmer-v3
RAD	**5107**$\pm$205	**9559**$\pm$610	**4402**$\pm$301	**100**$\pm$35
ADAM	4860$\pm$647	9417$\pm$872	4014$\pm$328	93$\pm$38
HB	924$\pm$19	2313$\pm$1465	837$\pm$1015	2$\pm$22
DLPF	917$\pm$16	2038$\pm$371	1722$\pm$1333	24$\pm$24
RGD	912$\pm$37	1383$\pm$806	765$\pm$355	15$\pm$29
NAG	887$\pm$46	1296$\pm$735	478$\pm$187	11$\pm$26
SGD	920$\pm$40	2372$\pm$735	DIV	1$\pm$20
$\Uparrow$	5.1%	1.5%	9.7%	7.5%

B. *Autonomous driving tasks*

To thoroughly assess RAD's performance in complex real-world tasks, we employ the IDSim benchmark for autonomous driving tests at simulated urban intersections [36]. We generate 20 intersections with varying typologies, each featuring mixed traffic flows comprising cars, bicycles, and pedestrians. To replicate the real-world perception conditions where noise is unavoidable, we deliberately add Gaussian noise to the observations of other road users, as illustrated in Table IV. Additionally, we implement a traffic signal system, where left turns and straight going are controlled by a single signal, creating challenging driving situations such as unprotected left turns. The objective of the task is to maneuver the ego vehicle through intersections by controlling its acceleration and steering angle within a limited number of control steps.

TABLE IV: Standard deviations of observation noise. All noises obey standard Gaussian distributions derived from real-world datasets.

Observation	Car	Bicycle	Pedestrian
Longitudinal position (m)	4.34×10^{-3}	1.75×10^{-3}	4.53×10^{-3}
Lateral position (m)	6.04×10^{-3}	1.95×10^{-3}	4.80×10^{-3}
Velocity (m/s)	7.51×10^{-2}	2.64×10^{-2}	6.58×10^{-2}
Heading angle (rad)	5.32×10^{-3}	2.17×10^{-3}	3.65×10^{-2}

We compare RAD with the widely-used ADAM optimizer within the context of the ADP algorithm [37]. After convergence, we test the two policies at each intersection 50 times to assess their driving performance. Table V presents five evaluation metrics: the success rate, collision rate, failure rate, driving comfort, and travel efficiency. The success rate represents the ratio of successful navigation through the intersection, while the collision rate measures instances of collisions with other road users. The failure rate encompasses cases where the ego vehicle either goes out of the drivable area or exceeds the maximum control steps. Driving comfort is evaluated by the root mean square of acceleration and yaw rate, and travel efficiency denotes the relative ratio between ego speed and road speed limits.

The results demonstrate that the policy trained with RAD achieves a remarkable success rate of 93.2%, significantly outperforming ADAM by a margin of 3.7%. Specifically, The policy trained with RAD exhibits lower collision rates, improved driving comfort, and higher travel efficiency than ADAM. Due to insufficient long-term training stability, the policy trained with ADAM experiences difficulties in accurately interpreting complex traffic conditions at intersections, leading to inadequate control commands and unsatisfactory interactions with other traffic participants. On the other hand, RAD stabilizes the training process by reducing the effects of noisy policy gradients arising from the intricate driving environment. Consequently, RAD produces more effective and reliable driving policies for autonomous vehicles. This experiment validates that RAD's applicability extends beyond standard RL benchmarks, such as MuJoCo and Atari, to real-world applications like autonomous driving. It highlights the potential of RAD as a robust optimization algorithm for complex, real-world tasks. Additionally, it is essential to note that ADP represents a model-based RL algorithm, which significantly diverges from the model-free nature of previously employed algorithms such as DDPG [10], SAC [12], TD3 [4], and DQN [8]. This further underscores the scalability of RAD across various RL settings.

TABLE V: Driving performance at urban intersections. Better metrics are highlighted in bold.

Optimizer	Success rate↑	Collision rate↓	Failure rate↓	Driving comfort↓	Travel efficiency↑
RAD	**93.2%**	**2.9%**	**3.9%**	**0.78**	**0.73**
ADAM	89.5%	5.8%	4.7%	1.00	0.68

S.IV Experimental settings

The detailed experimental settings are shown in Table VI and Table VII.

TABLE VI: Experimental settings I

Task	CartPole-v1	Hopper-v3	Other MuJoCo tasks
RL algorithm	DDPG & SAC	SAC	SAC
Discount factor	0.99	0.99	0.99
Exploration noise	$\varepsilon \sim \mathcal{N}(0, 0.1)$	/	/
Temperature coefficient	0.2	0.2	0.2
Approximate function	MLP	MLP	MLP
FC layer size	256×256	256×256	256×256
Activation function	ReLU	ReLU	ReLU
Learning rate decay	/	CosineAnnealingLR	/
Critic learning rate	5×10^{-4}	$1 \times 10^{-3} \rightarrow 1 \times 10^{-4}$	1×10^{-3}
Actor learning rate	5×10^{-5}	$1 \times 10^{-3} \rightarrow 1 \times 10^{-4}$	1×10^{-3}
Target network learning rate	5×10^{-3}	5×10^{-3}	5×10^{-3}
Maximum iteration	3×10^{4}	5×10^{5}	1×10^{6}
Batch size	1024	256	256
First-order momentum coeff.	0.9	0.9	0.9
Second-order momentum coeff.	0.999	0.999	0.999

TABLE VII: Experimental settings II

Task	MuJoCo tasks	Atari games	IDSim
RL algorithm	TD3	DQN	ADP
Discount factor	0.99	0.99	0.95
Exploration noise	$\varepsilon \sim \mathcal{N}(0, 0.1)$	ε-greedy policy	/
Approximate function	MLP	CNN	MLP
Number of conv. layers	/	3	/
FC layer size	256×256	512	256^5
Activation function	ReLU	ReLU	ReLU
Learning rate decay	/	/	LinearLR
Critic learning rate	3×10^{-4}	1×10^{-4}	$2 \times 10^{-5} \rightarrow 0$
Actor learning rate	3×10^{-4}	/	$2 \times 10^{-5} \rightarrow 0$
Target network learning rate	5×10^{-3}	/	/
Maximum iteration	1×10^{6}	1×10^{6}	3×10^{5}
Batch size	32	256	256
First-order momentum coeff.	0.9	0.9	0.9
Second-order momentum coeff.	0.999	0.999	0.999