A. Results in the Batch Setting

The proofs of convergence for many of the methods require independent samples for the updates. This condition is not generally met in the fully online learning setting that we consider throughout the rest of the paper. In Figure 7 we show results for all methods in the fully offline batch setting, demonstrating that—on the small problems that we consider—the conclusions do not change when transferring from the batch setting to the online setting. We include two additional methods in the batch setting, the Kernel Residual Gradient methods (Feng, Li & Liu, 2019), which do not have a clear fully online implementation.

We create a new batch dataset for each of 500 independent runs by getting 100k samples from the state distribution induced by the behavior policy, then sampling from the transition kernel for each of these states. We then perform mini-batch updates by sampling 8 independent transitions from this dataset. Each algorithm makes n updates for $n \in [1, 2, 4, 8, \dots, 8192]$, choosing the stepsize which minimizes the area under the RMSPBE learning for each n. This effectively shows the best performance of each algorithm if it was given a budget of n updates, allowing us to make comparisons across several different timescales. The constant stepsizes swept are $\alpha \in \{2^{-8}, 2^{-7}, \dots, 2^{0}\}$.

In Figure 7, we demonstrate that GTD2 and the Kernel-RG methods generally perform poorly across these set of domains. We additionally show that TDC, TD, and TDRC are often indistinguishable in the batch setting—except Boyan's Chain where TDC still performs inexplicably poorly—suggesting that perhaps TDRC's gain in performance of TDC is due to the correlated sampling induced by online learning. We finally show that TDC++, which is TDC with regularized $\mathbf{C}$, generally performs comparably to GTD2.

A.1. Relationship to Residual Gradients

The Residual Gradient (RG) family of algorithms provide an alternative gradient-based strategy for performing temporal difference learning. The RG methods minimize the Mean Squared Bellman Error (MSBE), while the Gradient TD family of algorithms minimize a particular form of the MSBE, the Mean Squared *Projected* Bellman Error (MSPBE). The RG family of methods generally suffer from difficulty in obtaining independent samples from the environment, leading towards stochastic optimization algorithms which find a biased solution (Sutton & Barto, 2018). However, very recent work has generalized the MSBE and proposed an algorithmic strategy to perform unbiased stochastic updates (Feng, Li & Liu, 2019; Dai et al., 2018). We compare to the approach in Feng, Li, and Liu (2019) below.

A.2. Derivation of the TDC++ Update Equations

In this section, we derive the update equations for TDC++, i.e. TDC with the regularized $\mathbf{C}_\beta$ matrix. Consider the MSPBE objective (see Eq. 7) but with a regularized $\mathbf{C}_\beta$:

$$\begin{aligned}\text{MSPBE}_{\text{++}}(\mathbf{w}_t) &\overset{\text{def}}{=} \mathbb{E}[\delta_t \mathbf{x}_t]^\top \left(\mathbb{E}\left[\mathbf{x}_t \mathbf{x}_t^\top\right]^{-1} + \beta\,\mathbf{I}\right) \mathbb{E}[\delta_t \mathbf{x}_t] \\ &= (-\mathbf{A}\mathbf{w} + \mathbf{b})^\top \mathbf{C}_\beta^{-1} (-\mathbf{A}\mathbf{w} + \mathbf{b}).\end{aligned}$$

The gradient of this objective is $-\frac{1}{2}\nabla_{\mathbf{w}} \text{MSPBE}_{\text{++}}(\mathbf{w}_t) = \mathbf{A}^\top \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\,\mathbf{w}_t) = \mathbb{E}[\delta_t \mathbf{x}_t] - \gamma \mathbb{E}\left[\mathbf{x}' \mathbf{x}^\top\right] \mathbf{h}_\beta - \beta\,\mathbf{h}_\beta$. Using this gradient and the same update for $\mathbf{h}_{t+1}$ as in TDRC, we obtain the update equations for TDC++ (with an additional η in the stepsize for $\mathbf{h}$):

$$\begin{aligned}\mathbf{h}_{t+1} &\leftarrow \mathbf{h}_t + \eta\alpha\left[\delta_t - (\mathbf{h}_t^\top \mathbf{x}_t)\right]\mathbf{x}_t - \eta\alpha\beta\mathbf{h}_t \\ \mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha\delta_t \mathbf{x}_t - \alpha\gamma(\mathbf{h}_t^\top \mathbf{x})\mathbf{x}_{t+1} - \alpha\beta\mathbf{h}_t.\end{aligned}$$

A.3. Convergence of TDC++

It is straightforward to show that TDC++ converges to the TD fixed point under very similar conditions as TDC (Maei, 2011). We show the key steps here (for details see Maei (2011) or Appendix H). The $\mathbf{G}$ matrix for TDC++ is $\mathbf{G} = \begin{bmatrix} -\eta\,\mathbf{C}_\beta & -\eta\,\mathbf{A} \\ \mathbf{A}^\top - \mathbf{C}_\beta & -\mathbf{A} \end{bmatrix}$. If we can show that the real parts of all the eigenvalues of $\mathbf{G}$ are negative, then the algorithm would converge. First note that for an eigenvalue $\lambda \in \mathbb{C}$ of $\mathbf{G}$, $\det(\mathbf{G} - \lambda\,\mathbf{I}) = \det(\lambda(\mathbf{C}_\beta + \lambda\,\mathbf{I}) + \mathbf{A}(\eta\,\mathbf{A}^\top + \lambda\,\mathbf{I})) = 0$. Then for some non–zero vector $\mathbf{z} \in \mathbb{C}$, $\mathbf{z}^*(\lambda(\mathbf{C}_\beta + \lambda\,\mathbf{I}) + \mathbf{A}(\eta\,\mathbf{A}^\top + \lambda\,\mathbf{I}))\,\mathbf{z} = 0$. Upon simplifying this, we obtain the following quadratic equation in λ:

$$\|\,\mathbf{z}\,\|^2 \lambda^2 + (\mathbf{z}^*(\eta\,\mathbf{C}_\beta + \mathbf{A})\,\mathbf{z})\lambda + \eta\|\,\mathbf{A}\,\mathbf{z}\,\|^2 = 0.$$

If λ_1 and λ_2 are two solutions of this equation, then

$$\lambda_1 \lambda_2 = \eta \frac{\|\,\mathbf{A}\,\mathbf{z}\,\|^2}{\|\,\mathbf{z}\,\|^2}, \qquad \lambda_1 + \lambda_2 = -\frac{(\mathbf{z}^*(\eta\,\mathbf{C}_\beta + \mathbf{A})\,\mathbf{z})}{\|\,\mathbf{z}\,\|^2}.$$

Since, $\lambda_1 \lambda_2 > 0$ and real, the real parts of both λ_1 and λ_2 have the same sign. Thus, $\text{Re}(\lambda_1 + \lambda_2) < 0$ would imply that each of $\text{Re}(\lambda_1) < 0$ and $\text{Re}(\lambda_2) < 0$ and we would be done. Assuming $\text{Re}(\lambda_1 + \lambda_2) = -\frac{(\mathbf{z}^*(\eta\,\mathbf{C}_\beta + \mathbf{A})\,\mathbf{z})^* + (\mathbf{z}^*(\eta\,\mathbf{C}_\beta + \mathbf{A})\,\mathbf{z})}{2\|\,\mathbf{z}\,\|^2} = -\frac{\mathbf{z}^*(\eta\,\mathbf{C}_\beta + \mathbf{H})\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2} < 0$, where $\mathbf{H} \overset{\text{def}}{=} \frac{1}{2}(\mathbf{A} + \mathbf{A}^\top)$, leads to the condition

$$\eta > -\lambda_{\text{min}}(\mathbf{C}_\beta^{-1}\,\mathbf{H}),$$

for TDC++ to converge.

TDC++ differs from TDRC in that it has an extra term $(-\alpha\beta\,\mathbf{h}_t)$ in the update for the weight $\mathbf{w}_{t+1}$. Further, unlike TDRC, the convergence of TDC++ doesn't require any conditions on β.

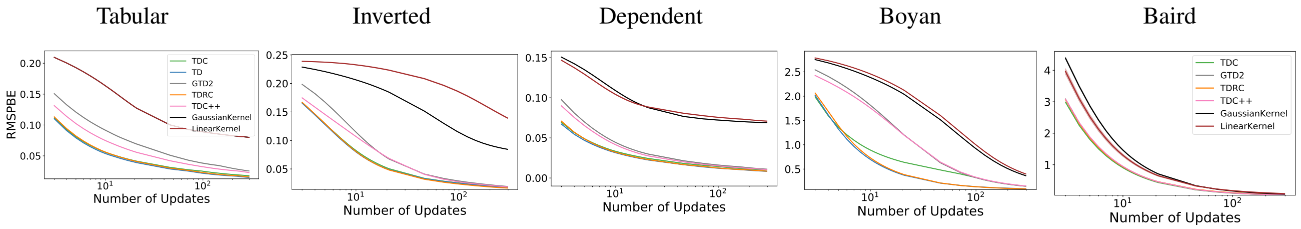

Figure 7. Sensitivity to the number of update steps for the offline batch setting. Each problem used a dataset of 100k samples sampled from the stationary distribution, then mini-batch updates used 8 independent samples from the dataset. On the x-axis we show a log-scale number of updates for each algorithm, on the y-axis we show the area under the RMSPBE learning curve averaged over 500 independent runs and 500 independently sampled datasets, with shaded regions showing the standard error over runs. For each number of update steps shown, we sweep over stepsizes and select the best stepsize for that number of updates; stepsizes were swept from $\alpha \in \{2^{-5}, 2^{-4}, \ldots, 2^0\}$. For TDRC, we set $\beta = 1$. This effectively shows the best performance of each algorithm if it was only given a fixed number of updates. GTD2 and the Kernel-RG methods show notably slower convergence than other methods.

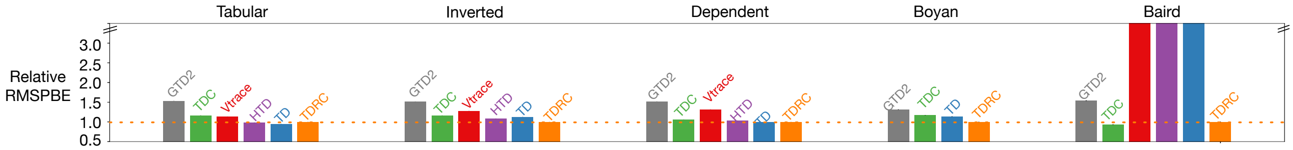

Figure 8. Relative performance of methods using the **Adam** stepsize selection algorithm, compared using the average area under the RMSPBE learning curve. Values swept are: $\alpha \in \{2^{-8}, 2^{-7}, \ldots, 2^{-1}, 2^0\}$ and as before, we set $\beta = 1$ for TDRC. On Baird's counterexample, TD, HTD, and VTrace all exhibit slow learning as well. The actual number for area under the learning curve are shown in Table 2.

B. Incorporating Accelerations

True stochastic gradient methods provide the benefit that they should be amenable to accelerations for stochastic approximation, such as momentum, mirror-prox updates (Juditsky & Nemirovski, 2011), and variance reduction techniques (Du et al., 2017). This is in fact one of the arguments motivating GTD2, and its formulation as a saddlepoint method.

We begin investigating how acceleration in the online prediction setting impacts the overall performance and relative ordering of the algorithms. Momentum is commonly used in online deep RL systems, and is a form of acceleration. We compare all the methods using Adam (Kingma & Ba, 2014; Reddi, Kale & Kumar, 2019), which includes momentum. Several recently proposed optimizers include momentum and are best viewed as extensions of Adam. Here we use Adam as there is little evidence in the literature that these new variants are better than Adam for online updates. We sweep over values of the meta-parameters in Adam, $\beta_1, \beta_2 \in \{0.9, 0.99, 0.999\}$, and select the values that best minimize the total RMSPBE separately for each algorithm.

The bar plot in Figure 8 parallels Figure 1, which uses Adagrad, with similar conclusions. The only notable difference is that TDC's performance on Boyan's chain is much better, though it is still not as good as TD and TDRC. Overall, the use of momentum did not accelerate convergence, with performance similar to Adagrad. The comparison is not perfect, as Adagrad allows the stepsizes to decrease to zero, which enables the algorithms to converge nicely on these domains. Adam does not due to the exponential average in the squared gradient term. These results, then, mainly provide a sanity check that results under an alternative optimizer are consistent with the previous results.

The majority of accelerations that can be used in policy evaluation are designed for off-line batch updates. Although we are more concerned with online performance, we use the batch setting in Appendix A as a sanity check to ensure that none of the recently proposed accelerated policy evaluation methods significantly outperform TD, TDC, or TDRC. In addition we include Kernel Residual Gradient (Kernel-RG) (Feng, Li & Liu, 2019). Figure 7 shows the performance of several methods given a fixed budget number of updates. Surprisingly, the Kernel-RG methods show much slower convergence across all problems tested.

C. Sensitivity to the Scale of $\mathbf{h}$

In Figure 3 we demonstrate TDRC's sensitivity to the regularization weight, β, which is responsible for balancing between the loss due to the regularizer and the mean-squared error for $\mathbf{h}$. We motivate empirically that, on a set of small domains, the scale of the regularizer does not significantly affect the performance of TDRC. However, as the scale of $\mathbf{h}$

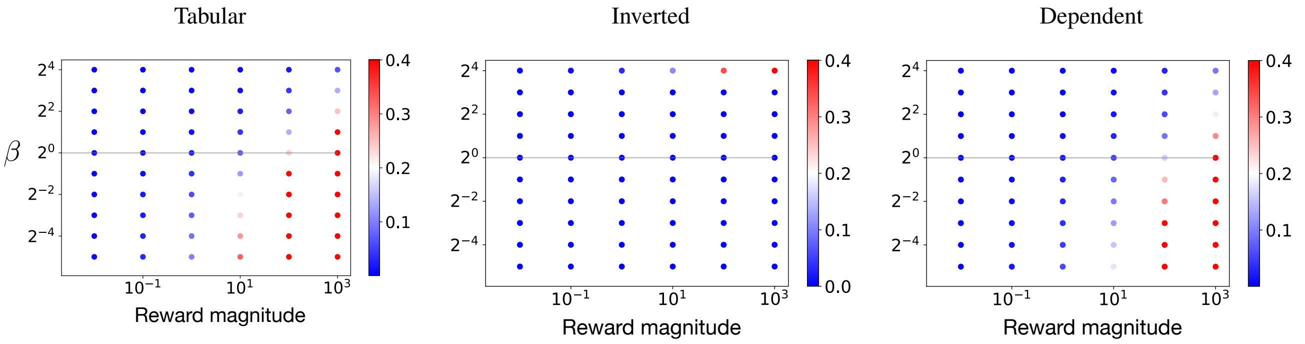

Figure 9. Relationship between TDRC and TD performance across different reward scales for different values of beta. On the x-axis we show the scale of the rewards for the terminal states of the random walk, on the y-axis we show a range of values of β. Each dot represents the number of standard deviations away from TD that TDRC's performance is across 500 independent runs for that particular value of β. For each dot, TDRC and TD choose the stepsize with lowest area under the RMSPBE learning curve; with stepsizes swept from $\alpha \in \{2^{-5}, 2^{-4}, \ldots, 2^{0}\}$. As the scale of the rewards increases (left to right on the x-axis), the variance of the secondary weights, $\mathbf{h}$, also increases; effectively requiring a larger value of β. This figure demonstrates that TDRC with $\beta = 1$ remains relatively insensitive to the scale of the rewards except in extreme cases when the variance of the rewards from transition to transition is quite large.

varies we likewise expect the scale of β to vary accordingly.

We design a set of small experiments to understand how changes in the environment cause the scale of $\mathbf{h}$ to change, and how that relates to the performance of TDRC across several values of β. The scale of $\mathbf{h}$ changes whenever the size of the TD error or scale of the features change. For these experiments, we chose to increase the range of the TD error by making the initial value function $V = \mathbf{0}$ and manipulating the magnitude of the rewards. We run this experiment on the five state random-walk domain with each of the feature representations used in Section 4, and change only the rewards in the terminal states by a multiplicative constant. We compute the mean and standard deviation of TD's performance across 500 independent runs and compute the number of standard deviations TDRC's mean performance is from TD's mean performance. We let the reward vary by order of magnitudes, with the multiplicative constant taking values $\{10^{-2}, 10^{-1}, \ldots, 10^{3}\}$. For each scaling, we test multiple values of $\beta \in \{2^{-5}, 2^{-4}, \ldots, 2^{4}\}$ and for each of these instances we select the best constant stepsize from $\{2^{-5}, 2^{-4}, \ldots, 2^{-1}\}$.

In Figure 9, we show the range of β for which TDRC's performance is as good, or nearly as good, as TD's performance as the magnitude of the rewards increases. As hypothesized, the range of acceptable β decreases as the reward magnitude increases; however, the range of β only appreciably shrinks for a pathologically large deviation between rewards and initial value function. This demonstrates that, while β is problem dependent, its range of acceptable values is robust to all but the most pathological of examples across several different representations.

D. Investigating QC on Mountain Car

In this section we include a deeper preliminary investigation into the performance of QC on the Mountain Car environment with non-linear function approximation. As we observed in Figure 5, QC performed considerably worse than either Q-learning and QRC. We hypothesize that this poor performance is the result of high variance updates to the value function estimate due to a poor estimate of $\mathbb{E}[\delta_t \mid S = s_t]$. We relax the restrictions on the secondary stepsize, $\eta\alpha$, by using $\eta = \frac{1}{2}$, allowing QC to become more like Q-learning and reducing the variance of the update to the secondary weights. We conclude by investigating the effects of prioritization of the replay buffer by drawing samples according to the squared TD error.

We start by investigating the performance of each algorithm when only a single step of replay is used on each environmental step. The learning curve in Figure 10 reaffirms that QRC and Q-learning significantly outperform QC in this setting. Interestingly, the norm of QC's secondary set of weights grows nearly monotonically throughout learning while in contrast, QRC's secondary weights start large at the beginning of learning and quickly shrink as the value function estimates become more accurate. The bottom right curve shows the mean and standard deviation of the maximum absolute value of $\hat{q}(S_t, \cdot)$ for each step of learning. The variance of QC's maximum state-action value increased significantly over the maximum observable return in the Mountain Car domain—which is represented by a dashed line at 100. These plots in combination suggest that QRC's additional constraint on the magnitude of the secondary weights helps stabilize the learning system when using neural network function approximators.

One plausible explanation for QC's poor performance is that

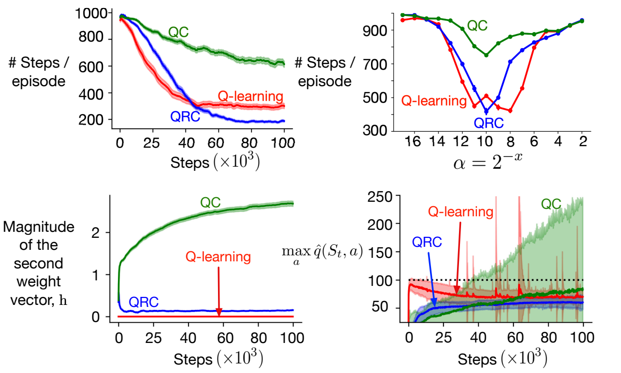

Figure 10. Control methods on Mountain Car with neural network function approximation. Each method takes one update step for every environment step and uses $\eta = 1$. **Top Left:** Average number of steps to goal. **Top Right:** Sensitivity to stepsize showing area under the learning curve for each value of α. **Bottom Left:** Magnitude of the secondary weights for each algorithm. Q-learning is included as a flat line at zero, as Q-learning is effectively a special case of QRC where the secondary weights are always $\mathbf{0}$. **Bottom Right:** Mean and standard deviation of the maximum action-value for each step of learning. QC exhibited massive growth in action-values throughout learning and Q-learning exhibited periodic spikes of instability.

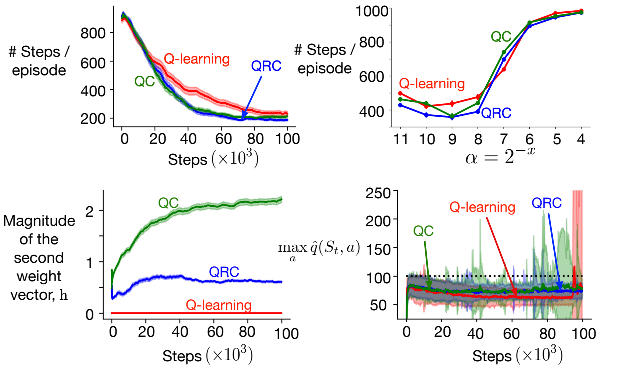

Figure 11. Same as Figure 10 except $\eta = 0.5$. Learning performance of QC is now competitive with Q-learning and QRC, though QC and Q-learning both exhibited more instability than QRC.

the TD error is high variance in the Mountain Car environment, increasing the variance of the stochastic updates to the secondary weights. We test this hypothesis by decreasing the stepsize for the secondary weights. If the variance of the updates is large, then a smaller stepsize can help stabilize learning. We choose $\eta = \frac{1}{2}$ and otherwise keep all other empirical settings the same.

Figure 11 shows that QRC and QC now perform very similarly and only slightly outperform Q-learning. As discussed in Section 4.2, decreasing the secondary stepsize makes both TDC and TDRC behave more similarly to TD, so this result is not surprising. Interestingly, Figure 11 shows that still the magnitude of the secondary weights quickly grows for QC; however, unlike the previous experiment, the secondary weights for QRC do not quickly decay either.

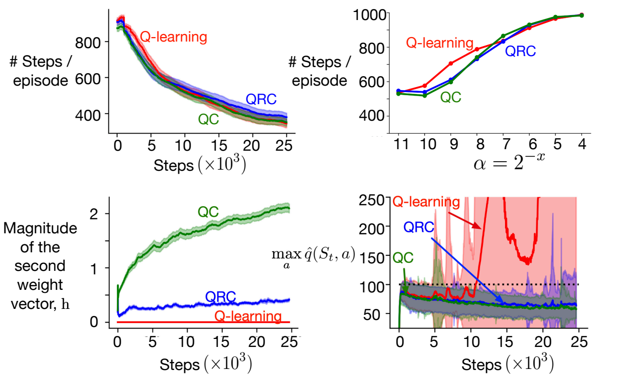

Figure 12. Same as Figure 10 except $\eta = 0.5$ and each method takes *ten* update steps for every environment step using prioritized experience replay.

Given that each of the algorithms seem to perform similarly when $\eta = \frac{1}{2}$, we revisit the highly off-policy experiment shown in Figure 5 when $\eta = \frac{1}{2}$. To further exaggerate the off-policy sampling, we additionally prioritize the experience replay buffer by drawing samples according to their squared TD error. Figure 12 shows that, while the learning curve performance between algorithms appears to be the same, Q-learning exhibits significant instability in its value function approximation.

These preliminary experiments suggest that, like TDC, QC's performance is highly driven by the magnitude of its secondary stepsize. When the secondary stepsize is well-tuned QC shows similar stability to QRC; while QRC remains stable across all experimental settings. Q-learning, like TD, is sensitive to the degree of off-policy data, becoming increasingly unstable as more off-policy updates are made. In each of the experimental settings included in this section, Q-learning exhibited occasional spikes of instability; further motivating the desire to extend sound Gradient TD methods for non-linear control.

E. Additional Linear Prediction Results

In this section we include additional results supporting the experiments run in the main body of the text. The primary conclusions drawn from these results were redundant with experiments in the text, but are included here for completeness.

We include results analogous to those in Section 4, except using a constant stepsize on all problems. While constant stepsizes are not commonly used in practice, they are useful

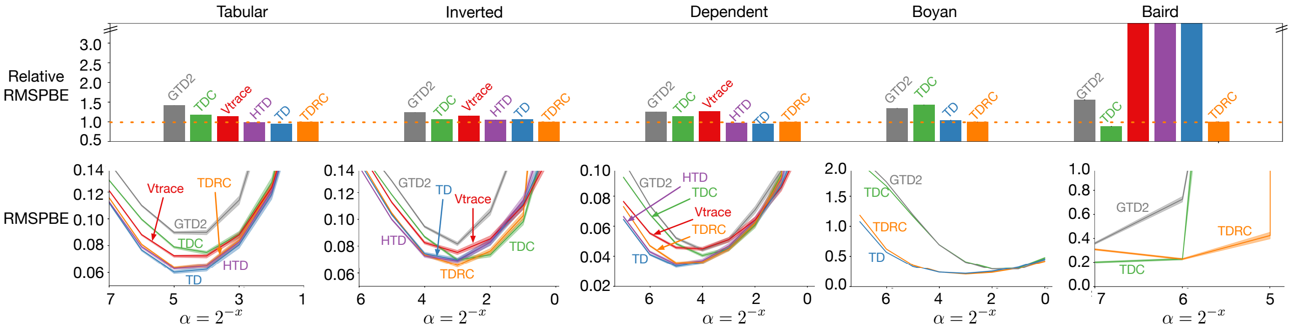

Figure 13. **Top:** The normalized average area under the RMSPBE learning curve for each method on each problem using a **constant** stepsize. Each bar is normalized by TDRC's performance so that each problem can be shown in the same range. All results are averaged over 200 independent runs with standard error bars shown at the top of each rectangle, though most are vanishingly small. **Bottom:** stepsize sensitivity measured using average area under the RMSPBE learning curve for each method on each problem. HTD and VTrace are not shown in Boyan's Chain because they reduce to TD for on-policy problems. The values corresponding to the bar graphs are given in Table 3.

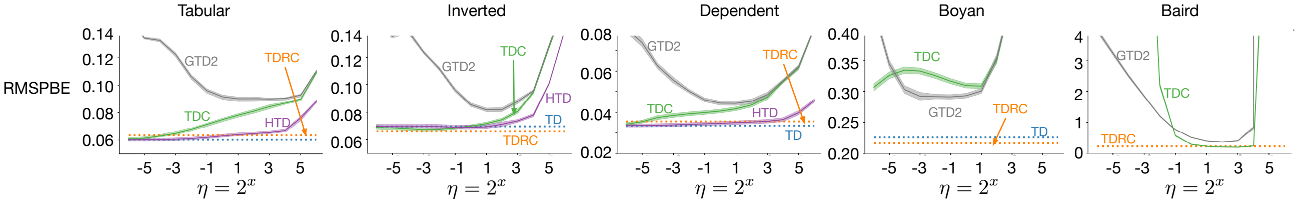

Figure 14. Sensitivity to the second stepsize, for changing parameter η. All methods use a constant stepsize α. All methods are free to choose any value of α for each specific value of η. Methods that do not have a second stepsize are shown as flat line.

for drawing clear conclusions without stepsize selection algorithm playing a confounding role. We show in Figure 13, that the relative performance between methods does not change when using a constant stepsize. We do notice that TDC performs more similarly to HTD, TD, and TDRC in the constant stepsize case, which suggests that TDC benefits less from using Adagrad than these other methods.

Figure 14 shows that algorithms are generally more similar in terms of stepsize sensitivity. This suggests that differences in between the algorithms are less pronounced when using constant stepsizes, which provides more support for the argument that empirical comparisons should simultaneously consider modern stepsize selection algorithms.

For completeness, we include the values visualized in Figure 1 as a table of values in Table 1. The standard error is reported for each entry in the table. The bold entries highlight the algorithm with the lowest RMSPBE for the given problem. The same is included for Figure 8 in Table 2 and for Figure 13 in Table 3.

F. Investigating Target Networks

One motivation for designing more stable off-policy algorithms is to improve learning interactions with neural network function approximators. A currently pervasive technique for improving stability of off-policy learning with neural networks is to use target networks. In this section, we investigate the impact of using target networks for each of the non-linear control algorithms investigated in this work.

In Figures 15, 16, and 17 we investigate the impact of synchronizing the target network to the value function approximation after every 4, 64, and 256 updates respectively. All the experimental settings remain the same, other than the rate of target network synchronization. The conclusions drawn in the main body of the paper continue to hold when using target networks; QC learns very slowly which is exaggerated by increasing delay in updates to the bootstrapped target, QRC is stable and insensitive to choice of stepsize, and Q-learning performs well but is negatively impacted by the introduction of target networks on these domains.

G. Parameter Settings and Other Experiment Details

G.1. Actor-Critic Algorithm with TDRC

We assume that the agent's policy $\pi_{\boldsymbol{\theta}}(A|S)$ is parameterized by weight vector $\boldsymbol{\theta}$. To incorporate TDRC into the one-step actor-critic algorithm (Sutton & Barto, 2018), we simply change the update rule for the value function approximation

	Tabular	Inverted	Dependent	Boyan	Baird
GTD2	0.079 ± 0.001	0.063 ± 0.001	0.041 ± 0.001	0.269 ± 0.003	0.357 ± 0.009
TDC	0.063 ± 0.001	0.053 ± 0.001	0.034 ± 0.001	0.639 ± 0.001	**0.196 ± 0.007**
HTD	0.048 ± 0.001	0.048 ± 0.001	0.025 ± 0.001	–	2.123 ± 0.013
TD	**0.046 ± 0.001**	0.051 ± 0.001	**0.024 ± 0.001**	0.248 ± 0.003	4.101 ± 0.095
VTrace	0.060 ± 0.001	0.059 ± 0.001	0.038 ± 0.001	–	4.101 ± 0.095
TDRC	0.049 ± 0.001	**0.047 ± 0.001**	0.026 ± 0.001	**0.222 ± 0.002**	0.242 ± 0.006

Table 1. Average area under the RMSPBE learning curve for each problem using the **Adagrad** algorithm. Bolded values highlight the lowest RMSPBE obtained for a given problem. These values correspond to the bar graphs in Figure 1.

	Tabular	Inverted	Dependent	Boyan	Baird
GTD2	0.094 ± 0.001	0.074 ± 0.001	0.048 ± 0.001	0.274 ± 0.006	0.356 ± 0.009
TDC	0.071 ± 0.002	0.057 ± 0.001	0.033 ± 0.001	0.244 ± 0.005	**0.215 ± 0.007**
HTD	0.060 ± 0.002	0.053 ± 0.001	0.032 ± 0.001	–	3.623 ± 0.027
TD	**0.058 ± 0.002**	0.055 ± 0.001	**0.031 ± 0.001**	0.237 ± 0.006	3.993 ± 0.053
VTrace	0.069 ± 0.001	0.063 ± 0.001	0.042 ± 0.001	–	3.993 ± 0.053
TDRC	0.061 ± 0.001	**0.049 ± 0.001**	0.031 ± 0.001	**0.209 ± 0.004**	0.232 ± 0.007

Table 2. Average area under the RMSPBE learning curve for each problem using the **Adam** stepsize selection algorithm. Bolded values highlight the lowest RMSPBE obtained for a given problem. These values correspond to the bar graphs in Figure 8.

	Tabular	Inverted	Dependent	Boyan	Baird
GTD2	0.090 ± 0.001	0.082 ± 0.001	0.044 ± 0.001	0.292 ± 0.004	0.361 ± 0.009
TDC	0.075 ± 0.001	0.070 ± 0.001	0.041 ± 0.001	0.309 ± 0.004	**0.205 ± 0.007**
HTD	0.063 ± 0.001	0.069 ± 0.002	0.035 ± 0.001	–	1184.368 ± 69.421
TD	**0.060 ± 0.001**	0.070 ± 0.002	**0.034 ± 0.001**	0.226 ± 0.005	11401.550 ± 270.628
VTrace	0.072 ± 0.001	0.076 ± 0.002	0.045 ± 0.001	–	18.239 ± 0.046
TDRC	0.064 ± 0.001	**0.066 ± 0.001**	0.036 ± 0.001	**0.217 ± 0.004**	0.232 ± 0.006

Table 3. Average area under the RMSPBE learning curve for each problem using the a **constant** stepsize. Bolded values highlight the lowest RMSPBE obtained for a given problem. These values correspond to the bar graphs in Figure 13.

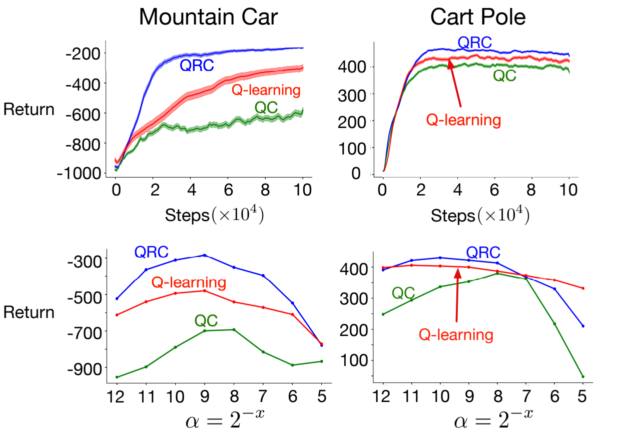

Figure 15. Non-linear control methods with target networks. Target network is synchronized with the value function after every 4 updates.

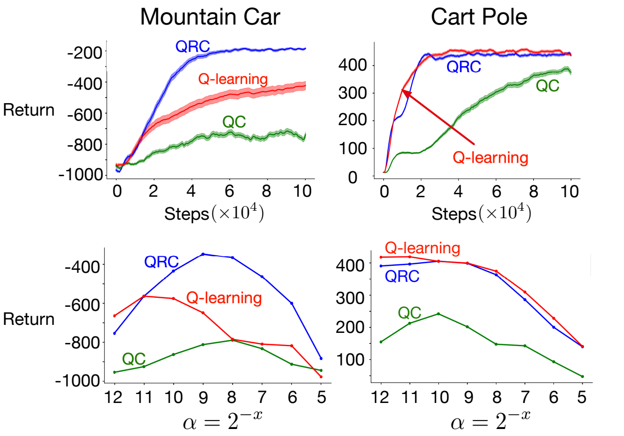

Figure 16. Same as Figure 15, except target network is synchronized after every 64 updates.

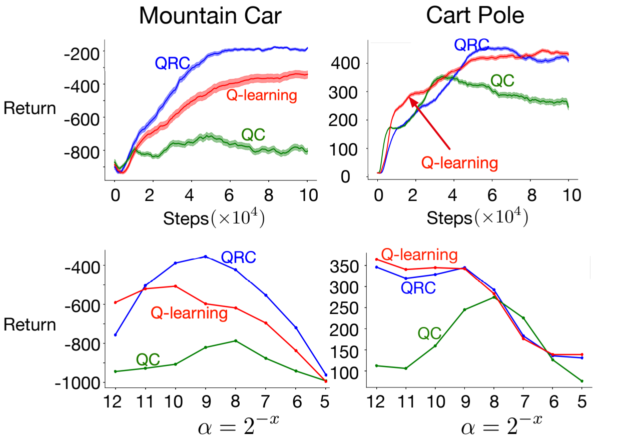

Figure 17. Same as Figure 15, except target network is synchronized after every 256 updates.

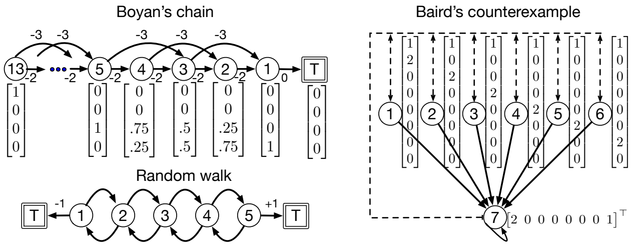

Figure 18. Above we provide a graphic depiction of each of the three MDPs and the corresponding feature representations used in our experiments. We omit the three feature representations used in the Random Walk due to space restrictions (see Sutton et al., 2009). All unlabeled transitions emit a reward of zero.

step for the TDRC update. This yields the following update equations for Actor-Critic with TDRC:

$$\begin{aligned}
\delta_t &= R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t \\
\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha \delta_t \mathbf{x}_t - \gamma (\mathbf{h}_t^\top \mathbf{x}_t) \mathbf{x}_{t+1} \\
\mathbf{h}_{t+1} &\leftarrow \mathbf{h}_t + \eta \alpha \left(\delta_t - \mathbf{h}_t^\top \mathbf{x}_t \right) \mathbf{x}_t - \eta \alpha \beta \mathbf{h}_t \\
\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t + \alpha \gamma^{t+1} \delta_t \nabla_{\boldsymbol{\theta}_t} \ln \pi_{\boldsymbol{\theta}}(A_t \mid S_t),
\end{aligned}$$

where the original actor-critic algorithm can be recovered with $\mathbf{h}_0 = \mathbf{0}$ and $\eta = 0$ and a TDC-based actor-critic algorithm can be obtained with $\beta = 0$. In practice, the γ^{t+1} term in the update for $\boldsymbol{\theta}$ is often dropped so, as such, in our actor-critic experiment we likewise did not include this term in our implementation. For ADAM optimizer we used $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We swept over $\alpha \in \{2^{-8}, 2^{-7}, \ldots, 2^{-2}, 2^{-1}\}$ and had $\eta = 1$ for TDC. We used tile coding with 5 tilings and 4×4 tiles.

G.2. Prediction Experimental Details

For the results shown in the main body of the paper on the random walk, Boyan's Chain, and Baird's Counterexample we swept over free meta-parameters for every method comparing the meta-parameters which performed best according to the area under the RMSPBE learning curve. The stepsizes swept for all algorithms were $\alpha \in \{2^{-7}, 2^{-6}, \ldots, 2^{0}\}$. For TDC and HTD, we swept values of the second stepsize by sweeping over a multiplicative constant times the primary stepsize, $\eta \in \{2^0, 2^1, \ldots, 2^6\}$ maintaining the convergence guarantees of the two-timescale proof of convergence for TDC. For GTD2, we swept values of $\eta \in \{2^{-6}, 2^{-5}, \ldots, 2^5, 2^6\}$ as the saddlepoint formulation of GTD2 allows for a much broader range of η while still maintaining convergence.

G.3. Cart Pole and Mountain Car Experimental Details

To solve these task we used a fully connected neural network with two hidden layers where each layer had 64 nodes in Cart Pole (32 nodes in Mountain Car) with ReLU as the non–linearity and the output layer as linear. The weights were updated using a replay buffer of size 4,096 in Cart Pole (size 4000 in Mountain Car) and mini-batch size of 32 using ADAM optimizer with $\epsilon = 10^{-8}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We also used ADAM optimizer for updating the $\mathbf{h}$ vector using $\epsilon = 10^{-8}$, $\beta_1 = 0.99$, and $\beta_2 = 0.999$. The neural network weights were initialized using Xavier initialization (Glorot & Bengio, 2010) and the biases were initialized with a normal distribution with mean 0 and standard deviation 0.1. The second weight vectors were initialized to $\mathbf{0}$. Actions were selected using an ϵ-greedy policy where $\epsilon = 0.1$. We tested several values of the stepsize: $\{2^{-13}, ..., 2^{-2}\}$ for Cart Pole and $\{2^{-17}, ..., 2^{-2}\}$ for Mountain Car. The final results show the performance averaged over 200 independent runs. In these task we set $\eta = 1$ for QC and QRC methods and set the regularization parameter $\beta = 1$ for QRC.

G.4. MinAtar Experimental Details

We ran the MinAtar experiments for 5 million steps. Discount factor parameter, γ was set to 0.99. The rewards were scaled by $(R \times (1 - \gamma))$ so that the neural network does not have to estimate large returns. The Q-Learning and QRC network architectures were the same as that used by (Young & Tian, 2019). The network had one convolutional layer and one fully connected layer after that. The convolutional layer used sixteen 3×3 convolutions with stride 1. The fully connected layer had 128 units. Both convolutional and fully connected layers used ReLU gates. The network is initialized the same way as (Young & Tian, 2019). We did not use target networks for MinAtar experiments because (Young & Tian, 2019) showed that using target networks has negligible effects on the results.

We used a circular replay buffer of size 100,000. The agent started learning when the replay buffer had 5,000 samples in it. We annealed epsilon from 1.0 to 0.1 through the first 100,000 steps and then kept it at 0.1 for the rest of the steps. The agent had one training step using a mini-batch of size 32 per environment step. As explained by (Young & Tian, 2019), frame skipping was not necessary since the frames of the MinAtar environment are more information rich. Other hyperparameters were chosen the same as (Young & Tian, 2019) and (Mnih et al., 2015). We used the RMSProp optimizer with a smoothing constant of 0.95, and $\epsilon = 0.01$. For QRC, we used RMSProp to learn the second weight vector $\mathbf{h}$. We swept over RMSprop stepsizes in powers of 2, $\{2^{-10}, ..., 2^{-5}\}$ for breakout, and $\{2^{-12}, ..., 2^{-8}\}$ for space invaders. η was set to 1 for QC and QRC and β was 1 for QRC.

For the learning curve, we plotted the setting that resulted in the best area under the learning curve. We computed the moving average of returns over 100 episodes (shown in Figure 6) similar to (Young & Tian, 2019). For computing the total discounted reward, we simply averaged over all of the returns that the agent got during 5 million steps to get a single number for each run and each parameter setting. We then averaged this number over 30 independent runs of the experiment to produce one point in the bottom part of Figure 6. For MinAtar experiments, we used python version 3.7, Pytorch version 1.4, and public code made available on Github for MinAtar[1].

H. Convergence of TDRC

In this section, we prove Theorem 3.1. Our analysis closely follows the one timescale proof for TDC convergence (Maei, 2011). We provide the full proof here for completeness.

H.1. Reformulating the TDRC Update

We combine the TDRC update equations (Eq. 8 and 9) into a single linear system in variable $\boldsymbol{\varrho}_t^\top \overset{\text{def}}{=} \left[\mathbf{h}_t^\top \, \mathbf{w}_t^\top\right]$:

$$\boldsymbol{\varrho}_{t+1} = \boldsymbol{\varrho}_t + \alpha_t(\mathbf{G}_{t+1}\, \boldsymbol{\varrho}_t + \mathbf{g}_{t+1}), \tag{12}$$

with $\mathbf{G}_{t+1} \overset{\text{def}}{=} \begin{bmatrix} -\eta(\mathbf{x}_t\, \mathbf{x}_t^\top + \beta\, \mathbf{I}) & \eta\rho_t\, \mathbf{x}_t(\gamma\, \mathbf{x}_{t+1} - \mathbf{x}_t)^\top \\ -\rho_t(\gamma\, \mathbf{x}_{t+1}\, \mathbf{x}_t^\top) & \rho_t\, \mathbf{x}_t(\gamma\, \mathbf{x}_{t+1} - \mathbf{x}_t)^\top \end{bmatrix}$ and $\mathbf{g}_{t+1} \overset{\text{def}}{=} \begin{bmatrix} \eta\rho_t R_{t+1}\, \mathbf{x}_t \\ \rho_t R_{t+1}\, \mathbf{x}_t \end{bmatrix}$.

For a random variable $\mathbf{X}$, using the definition of importance sampling, we know that $\mathbb{E}_b[\rho\, \mathbf{X}] = \mathbb{E}_\pi[\mathbf{X}]$. Further, while learning off–policy we assume the excursion setting and use the stationary state distribution corresponding to the behavior policy, i.e. $\mathbb{E}_\pi[\mathbf{x}_t\, \mathbf{x}_t^\top] = \sum_{S \in \mathcal{S}} d_b(S)\, \mathbf{x}(S)\, \mathbf{x}(S)^\top$, and consequently $\mathbb{E}_b[\mathbf{x}_t\, \mathbf{x}_t^\top] = \mathbb{E}_\pi[\mathbf{x}_t\, \mathbf{x}_t^\top]$. Therefore, $\mathbf{G} \overset{\text{def}}{=} \mathbb{E}_b[\mathbf{G}_k] = \begin{bmatrix} -\eta\, \mathbf{C}_\beta & -\eta\, \mathbf{A} \\ \mathbf{A}^\top - \mathbf{C} & -\mathbf{A} \end{bmatrix}$ and $\mathbf{g} \overset{\text{def}}{=} \mathbb{E}_b[\mathbf{g}_k] = \begin{bmatrix} \eta\, \mathbf{b} \\ \mathbf{b} \end{bmatrix}$, and Eq. 12 can be rewritten as

$$\boldsymbol{\varrho}_{t+1} = \boldsymbol{\varrho}_t + \alpha_t\big(h(\boldsymbol{\varrho}_t) + M_{t+1}\big), \tag{13}$$

where $h(\boldsymbol{\varrho}) \overset{\text{def}}{=} \mathbf{G}\, \boldsymbol{\varrho} + \mathbf{g}$ and $M_{t+1} \overset{\text{def}}{=} (\mathbf{G}_{t+1} - \mathbf{G})\, \boldsymbol{\varrho}_t + (\mathbf{g}_{t+1} - \mathbf{g})$ is the noise sequence. Also, let $\mathcal{F}_t \overset{\text{def}}{=} \sigma(\boldsymbol{\varrho}_1, M_1, \ldots, \boldsymbol{\varrho}_{t-1}, M_t)$.

H.2. Main Proof

To prove the convergence of TDRC, we use the results from Borkar & Meyn (2000) which require the following to be true: (i) The function $h(\boldsymbol{\varrho})$ is Lipschitz and there exists $h_\infty(\boldsymbol{\varrho}) \overset{\text{def}}{=} \lim_{c \to \infty} \frac{h(c\, \boldsymbol{\varrho})}{c}$ for all $\boldsymbol{\varrho} \in \mathbb{R}^{2d}$; (ii) The sequence $(M_t, \mathcal{F}_t)$ is a Martingale difference sequence (MDS), and $\mathbb{E}\left[\|M_{t+1}\|^2 \mid \mathcal{F}_t\right] \leq c_0(1 + \|\, \boldsymbol{\varrho}\, \|^2)$ for any initial parameter

[1] https://github.com/kenjyoung/MinAtar

Box 1: Derivation of Eq. 14.

Following the analysis given in Maei (2011), we write

$$\det(\mathbf{G}-\lambda\,\mathbf{I}) = \det\begin{bmatrix}-\eta\,\mathbf{C}_\beta-\lambda\,\mathbf{I} & -\eta\,\mathbf{A}\\ \mathbf{A}^\top-\mathbf{C} & -\,\mathbf{A}-\lambda\,\mathbf{I}\end{bmatrix} = (-1)^{2d}\det\begin{bmatrix}\eta\,\mathbf{C}_\beta+\lambda\,\mathbf{I} & \eta\,\mathbf{A}\\ \mathbf{C}-\mathbf{A}^\top & \mathbf{A}+\lambda\,\mathbf{I}\end{bmatrix}.$$

For a matrix $\mathbf{U}=\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\\ \mathbf{A}_3 & \mathbf{A}_4\end{bmatrix}$, $\det(\mathbf{U})=\det(\mathbf{A}_1)\cdot\det(\mathbf{A}_4-\mathbf{A}_3\,\mathbf{A}_1^{-1}\,\mathbf{A}_2)$. Further, since $\mathbf{C}$ is positive semi–definite, $\mathbf{C}_\beta+\lambda\,\mathbf{I}$ would be non–singular for any $\beta>0$. Using these results, we get

$$\det(\mathbf{G}-\lambda\,\mathbf{I}) = \det(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})\cdot\det(\mathbf{A}+\lambda\,\mathbf{I}-\eta(\mathbf{C}-\mathbf{A}^\top)(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}). \tag{B1}$$

Now $\eta\,\mathbf{C}\left(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\right)^{-1} = \Big(\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)-(\eta\beta+\lambda)\,\mathbf{I}\Big)\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1} = \mathbf{I}-(\eta\beta+\lambda)\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1}$. We can then write

$$\begin{aligned}
&\mathbf{A}+\lambda\,\mathbf{I}-\eta(\mathbf{C}-\mathbf{A}^\top)(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}\\
&=\mathbf{A}+\lambda\,\mathbf{I}-\eta\,\mathbf{C}(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}+\eta\,\mathbf{A}^\top(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}\\
&=\mathbf{A}+\lambda\,\mathbf{I}-\Big(\mathbf{I}-(\eta\beta+\lambda)\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1}\Big)\,\mathbf{A}+\eta\,\mathbf{A}^\top(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}\\
&=\lambda\,\mathbf{I}+(\eta\beta+\lambda)\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1}\,\mathbf{A}+\eta\,\mathbf{A}^\top(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})^{-1}\,\mathbf{A}\\
&=\Bigg[\lambda\,(\mathbf{A})^{-1}\left(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\right)+(\eta\beta+\lambda)\,\mathbf{I}+\eta\,\mathbf{A}^\top\Bigg]\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1}\,\mathbf{A}\\
&=(\mathbf{A})^{-1}\Bigg[\lambda\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)+\mathbf{A}\left(\eta\,\mathbf{A}^\top+(\eta\beta+\lambda)\,\mathbf{I}\right)\Bigg]\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)^{-1}\,\mathbf{A}\,.
\end{aligned}$$

Putting the above result in Eq. B1 along with the fact that $\det(\mathbf{A}_1\,\mathbf{A}_2)=\det(\mathbf{A}_1)\cdot\det(\mathbf{A}_2)$, we get

$$\det(\mathbf{G}-\lambda\,\mathbf{I}) = \det\Big(\lambda\big(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I}\big)+\mathbf{A}\left(\eta\,\mathbf{A}^\top+(\eta\beta+\lambda)\,\mathbf{I}\right)\Big).$$

vector $\boldsymbol{\varrho}_1$ and some constant $c_0>0$; (iii) The stepsize sequence α_t satisfies $\sum_t\alpha_t=\infty$ and $\sum_t\alpha_t^2<\infty$; (iv) The origin is a globally asymptotically stable equilibrium for the ODE $\dot{\boldsymbol{\varrho}}=h_\infty(\boldsymbol{\varrho})$; and (v) The ODE $\dot{\boldsymbol{\varrho}}=h(\boldsymbol{\varrho})$ has a unique globally asymptotically stable equilibrium.

The function $\mathbf{h}(\boldsymbol{\varrho})=\mathbf{G}\,\boldsymbol{\varrho}+\mathbf{g}$ is Lipschitz with the coefficient $\|\,\mathbf{G}\,\|$ and $\mathbf{h}_\infty(\boldsymbol{\varrho})=\mathbf{G}\,\boldsymbol{\varrho}$ is well defined for all $\boldsymbol{\varrho}\in\mathbb{R}^{2d}$. $(M_t,\mathcal{F}_t)$ is an MDS, since by construction it satisfies $\mathbb{E}[M_{t+1}\mid\mathcal{F}_t]=0$ and $M_t\in\mathcal{F}_t$. The coverage assumption implies that the second moments of ρ_t are uniformly bounded. Then applying triangle inequality to $M_{t+1}=(\mathbf{G}_{t+1}-\mathbf{G})\,\boldsymbol{\varrho}_t+(\mathbf{g}_{t+1}-\mathbf{g})$ and using the boundedness of second moments of the quadruplets $(\mathbf{x}_t,R_t,\mathbf{x}_{t+1},\rho_t)$, we get $\mathbb{E}\left[\|M_{t+1}\|^2\mid\mathcal{F}_t\right]\le\mathbb{E}\left[\|(\mathbf{G}_{t+1}-\mathbf{G})\,\boldsymbol{\varrho}_t\,\|^2\mid\mathcal{F}_t\right]+\mathbb{E}\left[\|\,\mathbf{g}_{t+1}-\mathbf{g}\,\|^2\mid\mathcal{F}_t\right]\le c_0(\|\,\boldsymbol{\varrho}_t\,\|^2+1)$. Condition on the stepsizes follows from our assumptions in the theorem statement. To verify the conditions (iv) and (v), we first show that the real parts of all the eigenvalues of $\mathbf{G}$ are negative.

H.3. Proving that the Real Parts of Eigenvalues of G are Negative (assuming C to be non–Singular)

In this section, we consider the case when the $\mathbf{C}$ matrix is non–singular. TDRC converges even when $\mathbf{C}$ is singular under alternate conditions, which are given in Section H.4. From Box 1, we obtain

$$\begin{aligned}\det(\mathbf{G}-\lambda\,\mathbf{I}) =\det\Big(&\lambda(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})\\ &+\mathbf{A}(\eta\,\mathbf{A}^\top+(\eta\beta+\lambda)\,\mathbf{I})\Big),\end{aligned}\tag{14}$$

for some $\lambda\in\mathbb{C}$. Now because an eigenvalue λ of matrix $\mathbf{G}$ satisfies $\det(\mathbf{G}-\lambda\,\mathbf{I})=0$, there must exist a non–zero vector $\mathbf{z}\in\mathbb{C}^d$ such that $\mathbf{z}^*[\lambda(\eta\,\mathbf{C}+(\eta\beta+\lambda)\,\mathbf{I})+\mathbf{A}(\eta\,\mathbf{A}^\top+(\eta\beta+\lambda)\,\mathbf{I})]\,\mathbf{z}=0$, which is equivalent to

$$\begin{aligned}\lambda^2+\left(\eta\beta+\eta\frac{\mathbf{z}^*\,\mathbf{C}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}+\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}\right)\lambda&\\ +\eta\left(\beta\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}+\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}\right)&=0.\end{aligned}$$

Box 2: Solutions of Eq. 15.

The solutions of a quadratic $ax^2+bx+c=0$ are given by $x=-\frac{b}{2a}\pm\frac{\sqrt{b^2-4ac}}{2a}$. Using this, we solve for λ in Eq. 15:

$$\begin{aligned} 2\lambda &= -(\eta\beta+\eta b_c+\lambda_z) \pm \sqrt{(\eta\beta+\eta b_c+\lambda_z)^2 - 4\eta(\beta\lambda_z+b_a)} \\ &= -\big(\eta\beta+\eta b_c+(\lambda_r+\lambda_c i)\big) \pm \sqrt{\big(\eta\beta+\eta b_c+(\lambda_r+\lambda_c i)\big)^2 - 4\eta\big(\beta(\lambda_r+\lambda_c i)+b_a\big)} \\ &= -\Omega - \lambda_c i \pm \sqrt{(\Omega+\lambda_c i)^2 - 4\eta(\beta\lambda_r+b_a) - 4\eta\beta\lambda_c i} \\ &= -\Omega-\lambda_c i \pm \sqrt{\big(\Omega^2-\lambda_c^2-4\eta(\beta\lambda_r+b_a)\big)+\big(2\Omega\lambda_c-4\eta\beta\lambda_c\big)i} \\ &= -\Omega-\lambda_c i \pm\sqrt{\big(\Omega^2-\Xi\big)+\big(2\Omega\lambda_c-4\eta\beta\lambda_c\big)i}, \end{aligned}$$

where in the second step we put $\lambda_z=\lambda_r+\lambda_c i$, and also we define $\Omega=\eta\beta+\eta b_c+\lambda_r$ and $\Xi=\lambda_c^2+4\eta(\beta\lambda_r+b_a)$, which are both real numbers.

We define $b_c=\frac{\mathbf{z}^*\,\mathbf{C}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}$, $b_a=\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}$, and $\lambda_z=\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}\equiv\lambda_r+\lambda_c i$ for $\lambda_r,\lambda_c\in\mathbb{R}$. The constants b_c and b_a are real and greater than zero for all non–zero vectors $\mathbf{z}$. Then the above equation can be written as

$$\lambda^2+(\eta\beta+\eta b_c+\lambda_z)\,\lambda+\eta(\beta\lambda_z+b_a)=0. \tag{15}$$

We solve for λ in Eq. 15 (see Box 2 for the full derivation) to obtain $2\lambda=-\Omega-\lambda_c i\pm\sqrt{(\Omega^2-\Xi)+(2\Omega\lambda_c-4\eta\beta\lambda_c)i}$, where we introduced intermediate variables $\Omega=\eta\beta+\eta b_c+\lambda_r$, and $\Xi=\lambda_c^2+4\eta(\beta\lambda_r+b_a)$, which are both real numbers.

Using $\mathrm{Re}(\sqrt{x+yi}) = \pm\frac{1}{\sqrt{2}}\sqrt{\sqrt{x^2+y^2}+x}$ we get $\mathrm{Re}(2\lambda)=-\Omega\pm\frac{1}{\sqrt{2}}\sqrt{\Upsilon}$, with the intermediate variable $\Upsilon=\sqrt{(\Omega^2-\Xi)^2+(2\Omega\lambda_c-4\eta\beta\lambda_c)^2}+(\Omega^2-\Xi)$. Next we obtain conditions on β and η such that the real parts of both the values of λ are negative for all non–zero vectors $\mathbf{z}\in\mathbb{C}$.

H.3.1. CASE 1

First consider $\mathrm{Re}(2\lambda)=-\Omega+\frac{1}{\sqrt{2}}\sqrt{\Upsilon}$. Then $\mathrm{Re}(\lambda)<0$ is equivalent to

$$\Omega>\frac{1}{\sqrt{2}}\sqrt{\Upsilon}. \tag{16}$$

Since, the right hand side of this inequality is clearly positive, we must have

$$\Omega=\eta\beta+\eta b_c+\lambda_r>0. \tag{C1}$$

This gives us our first condition on η and β. Simplifying Eq. 16 and putting back the values for the intermediate variables (see Box 3 for details), we get

$$\Omega^2+\Xi>\sqrt{(\Omega^2-\Xi)^2+(2\Omega\lambda_c-4\eta\beta\lambda_c)^2}. \tag{17}$$

Again, since the right hand side of the above inequality is positive, we must have

$$\Omega^2+\Xi=(\eta\beta+\eta b_c+\lambda_r)^2+\lambda_c^2+4\eta(\beta\lambda_r+b_a)>0. \tag{C2}$$

This is the second condition we have on η and β. Continuing to simplify the inequality in Eq. 17 (again see Box 3 for details), we get our third and final condition:

$$(\eta\beta+\eta b_c+\lambda_r)^2(\beta\lambda_r+b_a)+\beta\lambda_c^2(\eta b_c+\lambda_r)>0. \tag{C3}$$

If $\lambda_r>0$ for all $\mathbf{z}\in\mathbb{R}$, then each of the Conditions C1, C2, and C3 hold true and consequently TDRC converges. This case corresponds to the on–policy setting where the matrix $\mathbf{A}$ is positive definite and TD converges.

Now we show that TDRC converges even when $\mathbf{A}$ is not PSD (the case where TD is not guaranteed to converge). If we assume $\beta\lambda_r+b_a>0$ and $\eta b_c+\lambda_r>0$, then each of the Conditions C1, C2, and C3 again hold true and TDRC would converge. As a result we obtain the following bounds:

$$\beta<-\frac{b_a}{\lambda_r}\Rightarrow\beta<\min_{\mathbf{z}}\left(-\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{z}}{\mathbf{z}^*\,\mathbf{H}\,\mathbf{z}}\right), \tag{18}$$

$$\eta>-\frac{\lambda_r}{b_c}\Rightarrow\eta>\max_{\mathbf{z}}\left(-\frac{\mathbf{z}^*\,\mathbf{H}\,\mathbf{z}}{\mathbf{z}^*\,\mathbf{C}\,\mathbf{z}}\right), \tag{19}$$

with $\mathbf{H}\overset{\text{def}}{=}\frac{1}{2}(\mathbf{A}+\mathbf{A}^\top)$. These bounds can be made more interpretable. Using the substitution $\mathbf{y}=\mathbf{H}^{\frac{1}{2}}\,\mathbf{z}$ we obtain

$$\begin{aligned} \min_{\mathbf{z}}\left(-\frac{\mathbf{z}^*\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{z}}{\mathbf{z}^*\,\mathbf{H}\,\mathbf{z}}\right) &\equiv \min_{\mathbf{y}}\frac{\mathbf{y}^*(-\,\mathbf{H}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{H}^{-\frac{1}{2}})\,\mathbf{y}}{\|\,\mathbf{y}\,\|^2} \\ &= \lambda_{\text{min}}(-\,\mathbf{H}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{H}^{-\frac{1}{2}}) \\ &= -\lambda_{\text{max}}(\mathbf{H}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{A}^\top\,\mathbf{H}^{-\frac{1}{2}}) \\ &= -\lambda_{\text{max}}(\mathbf{H}^{-1}\,\mathbf{A}\,\mathbf{A}^\top), \end{aligned}$$

where λ_{max} represents the maximum eigenvalue of the matrix. Proceeding similarly for η, we can write the bounds in

Box 3: Simplification of Eq. 16.

Putting the value of $\Upsilon = \sqrt{(\Omega^2 - \Xi)^2 + (2\Omega\lambda_c - 4\eta\beta\lambda_c)^2} + (\Omega^2 - \Xi)$ back in $\Omega > \frac{1}{\sqrt{2}}\sqrt{\Upsilon}$, we get

$$
\begin{aligned}
& \Omega > \frac{1}{\sqrt{2}}\sqrt{\sqrt{(\Omega^2 - \Xi)^2 + (2\Omega\lambda_c - 4\eta\beta\lambda_c)^2} + (\Omega^2 - \Xi)} \\
\Leftrightarrow\quad & \Omega^2 > \frac{1}{2}\left[\sqrt{(\Omega^2 - \Xi)^2 + (2\Omega\lambda_c - 4\eta\beta\lambda_c)^2} + (\Omega^2 - \Xi)\right] && \text{[squaring both sides]} \\
\Leftrightarrow\quad & \Omega^2 + \Xi > \sqrt{(\Omega^2 - \Xi)^2 + (2\Omega\lambda_c - 4\eta\beta\lambda_c)^2} \\
\Leftrightarrow\quad & (\Omega^2 + \Xi)^2 > (\Omega^2 - \Xi)^2 + (2\Omega\lambda_c - 4\eta\beta\lambda_c)^2 && \text{[squaring both sides]} \\
\Leftrightarrow\quad & \Omega^2\Xi > (\Omega\lambda_c - 2\eta\beta\lambda_c)^2 \\
\Leftrightarrow\quad & \Omega^2(\lambda_c^2 + 4\eta(\beta\lambda_r + b_a)) > \Omega^2\lambda_c^2 + 4\eta^2\beta^2\lambda_c^2 - 4\eta\beta\lambda_c^2\Omega && \text{[putting } \Xi = \lambda_c^2 + 4\eta(\beta\lambda_r + b_a)\text{]} \\
\Leftrightarrow\quad & \Omega^2\eta(\beta\lambda_r + b_a) > \eta^2\beta^2\lambda_c^2 - \eta\beta\lambda_c^2\Omega \\
\Leftrightarrow\quad & (\eta\beta + \eta b_c + \lambda_r)^2(\beta\lambda_r + b_a) > \eta\beta^2\lambda_c^2 - \beta\lambda_c^2(\eta\beta + \eta b_c + \lambda_r) && \text{[putting } \Omega = \eta\beta + \eta b_c + \lambda_r\text{]} \\
\Leftrightarrow\quad & (\eta\beta + \eta b_c + \lambda_r)^2(\beta\lambda_r + b_a) > -\beta\lambda_c^2(\eta b_c + \lambda_r) \\
\Leftrightarrow\quad & (\eta\beta + \eta b_c + \lambda_r)^2(\beta\lambda_r + b_a) + \beta\lambda_c^2(\eta b_c + \lambda_r) > 0.
\end{aligned}
$$

Note that all these steps have full equivalence (especially the squaring operations in second and fourth step are completely reversible), because we explicitly enforce that $\Omega > 0$ and $\Omega^2 + \Xi > 0$ in Conditions C1 and C2 respectively. As a result, if we satisfy conditions C1, C2, and C3, $\text{Re}(2\lambda) = -\Omega + \frac{1}{\sqrt{2}}\sqrt{\Upsilon} < 0$ would be satisfied as well.

Eq. 18 and 19 equivalently as

$$\beta < -\lambda_{\max}(\mathbf{H}^{-1}\,\mathbf{A}\,\mathbf{A}^\top), \tag{20}$$

$$\eta > -\lambda_{\min}(\mathbf{C}^{-1}\,\mathbf{H}). \tag{21}$$

If these bounds are satisfied by η and β then the real parts of all the eigenvalues of $\mathbf{G}$ would be negative and TDRC will converge.

H.3.2. CASE 2

Next consider $\text{Re}(2\lambda) = -\Omega - \frac{1}{\sqrt{2}}\sqrt{\Upsilon}$. The second term is always negative and we assumed $\Omega > 0$ in Eq. C1. As a result, $\text{Re}(\lambda) < 0$ and we are done.

Therefore, we get that the real part of the eigenvalues of $\mathbf{G}$ are negative and consequently condition (iv) above is satisfied. To show that condition (v) holds true, note that since we assumed $\mathbf{A} + \beta\,\mathbf{I}$ to be non–singular, $\mathbf{G}$ is also non–singular; this means that for the ODE $\dot{\boldsymbol{\varrho}} = h(\boldsymbol{\varrho})$, $\boldsymbol{\varrho}^* = -\,\mathbf{G}^{-1}\,\mathbf{g}$ is the unique asymptotically stable equilibrium with $\bar{\mathbf{V}}(\boldsymbol{\varrho}) \overset{\text{def}}{=} \frac{1}{2}(\mathbf{G}\,\boldsymbol{\varrho} + \mathbf{g})^\top(\mathbf{G}\,\boldsymbol{\varrho} + \mathbf{g})$ as its associated strict Lyapunov function.

H.4. Convergence of TDRC when C is Singular

When $\mathbf{C}$ is singular, $b_c = \frac{\mathbf{z}^*\,\mathbf{C}\,\mathbf{z}}{\|\,\mathbf{z}\,\|^2}$ is no longer always greater than zero for an arbitrary vector $\mathbf{z}$. Consequently, if we explicitly set $b_c = 0$ we would get alternative bounds on η and β for which TDRC would converge. Putting $b_c = 0$ in Conditions C1, C2, and C3, we get

$$
\begin{aligned}
\eta\beta + \lambda_r &> 0, \\
(\eta\beta + \lambda_r)^2 + \lambda_c^2 + 4\eta(\beta\lambda_r + b_a) &> 0, \text{and} \\
(\eta\beta + \lambda_r)^2(\beta\lambda_r + b_a) + \beta\lambda_c^2\lambda_r &> 0.
\end{aligned}
$$

As before, we are concerned with the case when $\mathbf{A}$ is not PSD and thus $\lambda_r < 0$. Further, assume that $\beta\lambda_r + b_a > 0$ (this is the same upper bound on β as given in Eq. 18). We simplify the third inequality above to obtain the bound on η. As a result, we get the following bounds for β and η:

$$\beta < -\frac{b_a}{\lambda_r}, \qquad \eta > \frac{1}{\beta}\left(\sqrt{\frac{-\beta\lambda_c^2\lambda_r}{\beta\lambda_r + b_a}} - \lambda_r\right). \tag{22}$$

The bound on η automatically satisfies the first condition $\eta\beta + \lambda_r > 0$. Therefore, if β and η satisfy these bounds, TDRC converges even for a singular $\mathbf{C}$ matrix.

I. Fixed Points of TDRC

Theorem I.1 (Fixed Points of TDRC) *If $\mathbf{w}$ is a TD fixed point, i.e., a solution to $\mathbf{A}\mathbf{w} = \mathbf{b}$, then it is a fixed point for the expected TDRC update,*

$$\mathbf{A}_\beta^\top \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}) = \mathbf{0}.$$

Further, the set of fixed points for TD and TDRC are equivalent if $\mathbf{C}_\beta$ is invertible and if $-\beta$ does not equal to any of *the eigenvalues of* $\mathbf{A}$*. Note that* $\mathbf{C}_\beta$ *is always invertible if* $\beta > 0$*, and is invertible if* $\mathbf{C}$ *is invertible even for* $\beta = 0$*.*

Proof: To show equivalence, the first part is straightforward: when $\mathbf{A}\mathbf{w} = \mathbf{b}$, then $\mathbf{b} - \mathbf{A}\mathbf{w} = \mathbf{0}$ and so $\mathbf{A}_\beta^\top \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}) = \mathbf{0}$. This means that any TD fixed point is a TDRC fixed point. Now we simply need to show that under the additional conditions, a TDRC fixed point is a TD fixed point.

If $-\beta$ does not equal any of the eigenvalues of $\mathbf{A}$, then $\mathbf{A}_\beta = \mathbf{A} + \beta\mathbf{I}$ is a full rank matrix. Because both $\mathbf{A}_\beta$ and $\mathbf{C}_\beta$ are full rank, the nullspace of $\mathbf{A}_\beta^\top \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w})$ equals to the nullspace of $\mathbf{b} - \mathbf{A}\mathbf{w}$. Therefore, $\mathbf{w}$ satisfies $\mathbf{A}_\beta^\top \mathbf{C}_\beta^{-1}(\mathbf{b} - \mathbf{A}\mathbf{w}) = 0$ iff $(\mathbf{b} - \mathbf{A}\mathbf{w}) = \mathbf{0}$.

We can prove Theorem I.1, in an alternate fashion as well. The linear system in Eq. 12 has a solution (in expectation) which satisfies

$$\mathbf{G}\,\boldsymbol{\varrho} + \mathbf{g} = \mathbf{0}.$$

We show that this linear system has full rank and thus a single solution: $\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}$ and $\mathbf{h} = \mathbf{0}$. If we show that the matrix $\mathbf{G}$ is non–singular, i.e. its determinant is non–zero, we are done. From Eq. 14 it is straightforward to obtain

$$\det(\mathbf{G}) = \eta^{2d} \det(\mathbf{A}^\top + \beta\,\mathbf{I}) \cdot \det(\mathbf{A}),$$

which is non–zero if we assume that β does not equal the negative of any eigenvalue of $\mathbf{A}$ and that $\mathbf{A}$ is non–singular. ■