Finally, note that the quantity in brackets in the TD(0) update is a sort of error, measuring the difference between the estimated value of S_t and the better estimate $R_{t+1}+\gamma V(S_{t+1})$. This quantity, called the *TD error*, arises in various forms throughout reinforcement learning:

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \tag{6.5}$$

Notice that the TD error at each time is the error in the estimate *made at that time*. Because the TD error depends on the next state and next reward, it is not actually available until one time step later. That is, δ_t is the error in $V(S_t)$, available at time $t+1$. Also note that if the array V does not change during the episode (as it does not in Monte Carlo methods), then the Monte Carlo error can be written as a sum of TD errors:

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \qquad \text{(from (3.3))} \\
&= \delta_t + \gamma\big(G_{t+1} - V(S_{t+1})\big) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\big(G_{t+2} - V(S_{t+2})\big) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t-1}\delta_{T-1} + \gamma^{T-t}\big(G_T - V(S_T)\big) \\
&= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t-1}\delta_{T-1} + \gamma^{T-t}\big(0 - 0\big) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k.
\end{aligned} \tag{6.6}$$

This identity is not exact if V is updated during the episode (as it is in TD(0)), but if the step size is small then it may still hold approximately. Generalizations of this identity play an important role in the theory and algorithms of temporal-difference learning.

Exercise 6.1 If V changes during the episode, then (6.6) only holds approximately; what would the difference be between the two sides? Let V_t denote the array of state values used at time t in the TD error (6.5) and in the TD update (6.2). Redo the derivation above to determine the additional amount that must be added to the sum of TD errors in order to equal the Monte Carlo error.

Note that n-step returns for $n > 1$ involve future rewards and value functions that are not available at the time of transition from t to $t+1$. No real algorithm can use the n-step return until after it had seen R_{t+n} and computed V_{t+n-1}. The first time these are available to be used is $t+n$. The natural algorithm for using n-step returns is thus

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\left[G_t^{(n)} - V_{t+n-1}(S_t)\right], \qquad 0 \le t < T, \tag{7.3}$$

while the values of all other states remain unchanged, $V_{t+n}(s) = V_{t+n-1}(s), \forall s \neq S_t$. We call this algorithm *n-step TD*. Note that no changes at all are made during the first $n-1$ steps of each episode. To make up for that, an equal number of addition updates are made at the end of the episode, after termination and before starting the next episode. Complete pseudocode is given in the box on the next page.

Exercise 7.1 In Chapter 6 we noted that the Monte Carlo error can be written as the sum of TD errors if the value estimates don't change from step to step (6.6). Show that the n-step error used in (7.3) can also be written as a sum TD errors (again if the value estimates don't change) generalizing the earlier result.

Exercise 7.2 (programming) With an n-step method, the value estimates *do* change from step to step, so an algorithm that used the sum of TD errors (see previous exercise) in place of the error in (7.3) would actually be a slightly different algorithm. Would it be a better algorithm or a worse one? Devise and program a small experiment to answer this question empirically.