# HOMEWORK SOLUTIONS #6

1. (a) For the adaptive stepsize algorithm I used $\alpha = 0.0004$ and $\mu_{\max} = 0.1$, yielding average $\mu(n) =$`0.0372`. Recall that $\mu_{\text{opt}} =$ `0.0340`, indicating that this algorithm does quite a good job of adapting the stepsize to minimize steady-state MSE. Fig. 1 shows that the parameters track the true parameters quite well, and Fig. 2 shows that the steady-state error value is essentially optimal. Fig. 3 shows that the stepsize adapts quite quickly to a value near the optimal.

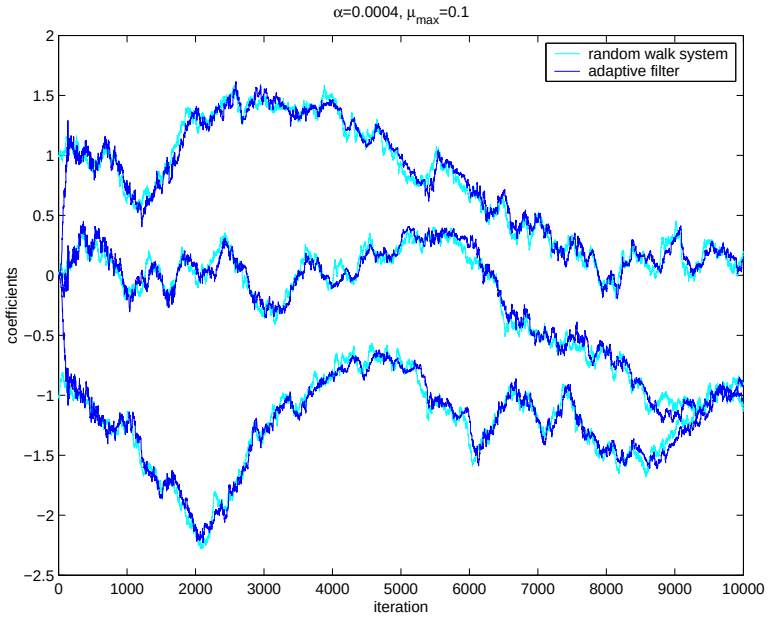


Figure 1: Adaptive-stepsize LMS: parameter trajectories.

(b) For the variable stepsize algorithm I used $\rho = 0.0005$ and $\mu_{\max} = 0.1$. This algorithm does not seem to perform as well as the previous: note from Fig. 4 that the trajectories sometimes adapt too slowly. Fig. 5 shows that the slow adaptation manifests itself as MSE bursts above the minimum. Examining Fig. 6, we find that $\mu(n)$ change erratically, and are often much smaller than optimal. My conclusion is that this algorithm, while computationally simple, does not appear to work very well.

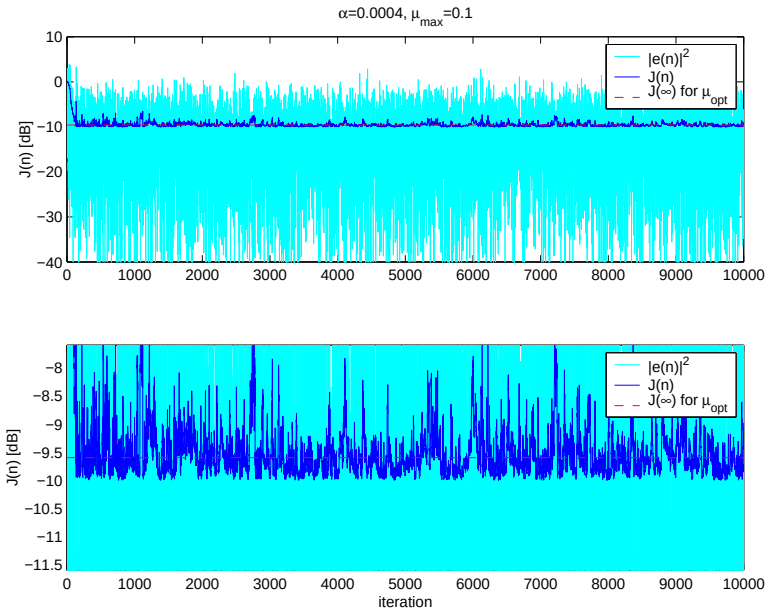


Figure 2: Adaptive-stepsize LMS: error trajectory.

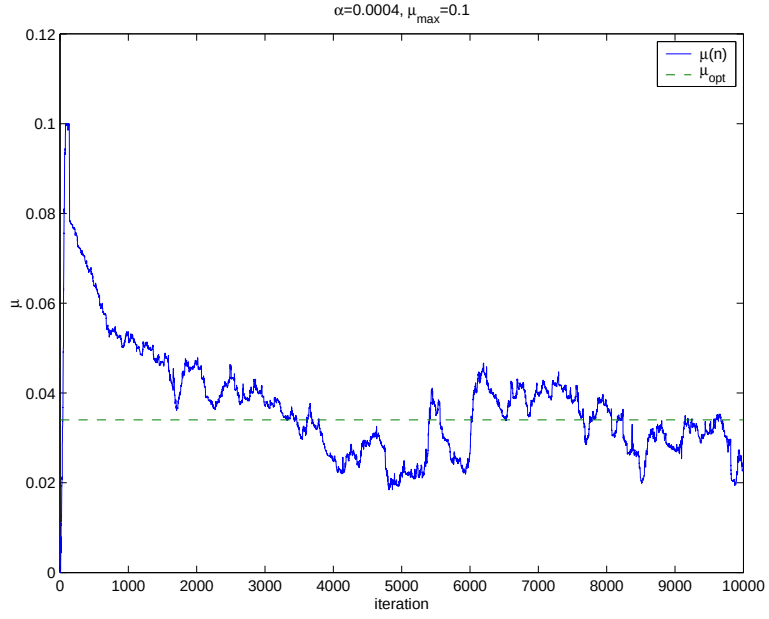


Figure 3: Adaptive-stepsize LMS: stepsize trajectory.

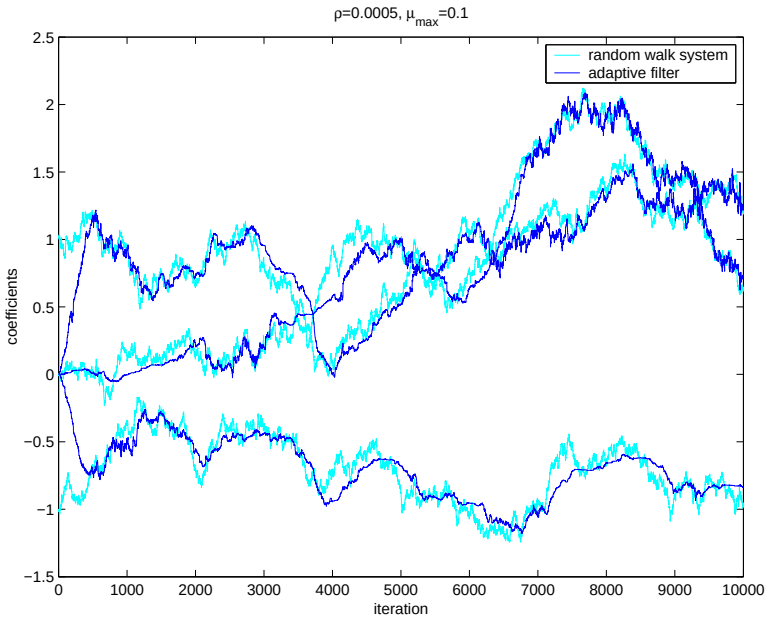


Figure 4: Variable-stepsize LMS: parameter trajectories.

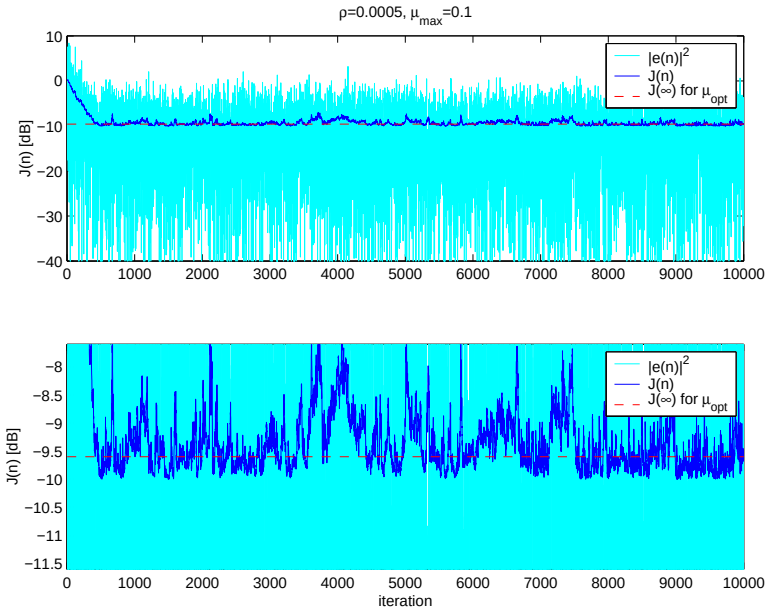


Figure 5: Variable-stepsize LMS: error trajectory.

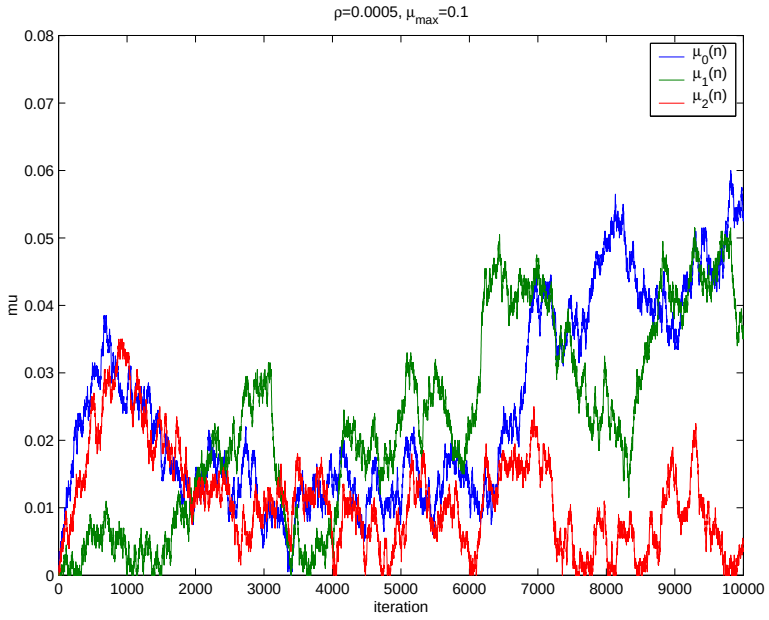


Figure 6: Variable-stepsize LMS: stepsize trajectories.

2. (a) The choice $\Delta = 5$ gives the lowest $J_{\text{mse}}$ as a function of $\Delta$. (See Fig. 7.)

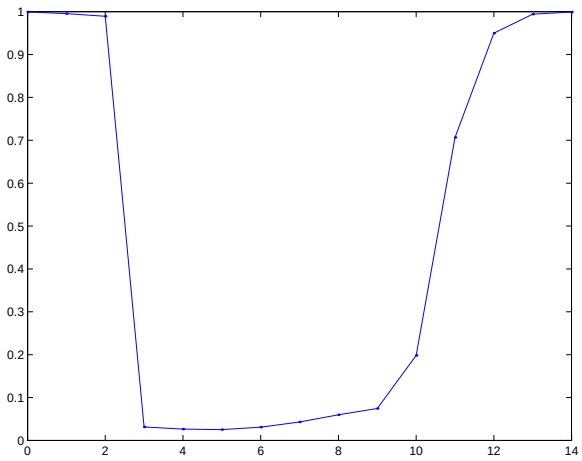


Figure 7: $J_{\min}$ versus $\Delta$.

(b) Fig. 8 compares SE-LMS to LMS. With equal steady-state error, the SE-LMS algorithm clearly takes longer to converge. Thus LMS can be considered to offer superior performance (at the cost of a more expensive implementation).

(c) Fig. 9 compares SR-LMS to LMS. Surprisingly, the SR-LMS algorithm converges only slightly slower than LMS (for equal steady-state MSE)!

(d) Fig. 10 compares DFT-based TDAF to LMS using a flat power spectrum initialization (i.e., $\hat{\boldsymbol{\lambda}}(0) = \mathbf{1}$). Though there is a slight improvement over LMS, the improvement may not be quite what we expect. This can be explained by looking at Fig. 11, which shows that $\hat{\boldsymbol{\lambda}}(n)$ has not adapted much from its unity initialization during the convergence stage of the adaptation. We know that when $\hat{\boldsymbol{\lambda}}(n) \approx \mathbf{1}$, the TDAF should behave nearly identical to LMS (since rotating the parameter space by a unitary matrix preserves the shape of the MSE contours.)

(e) If we initialize TDAF at the correct locations (which may not be possible in practice) then the convergence of TDAF can be significantly faster than that of LMS, as shown in Fig. 12. Using the same forgetting factor as in the previous experiment, we find that the values $\hat{\boldsymbol{\lambda}}(n)$ stay quite close to the optimal values (see Fig. 13).

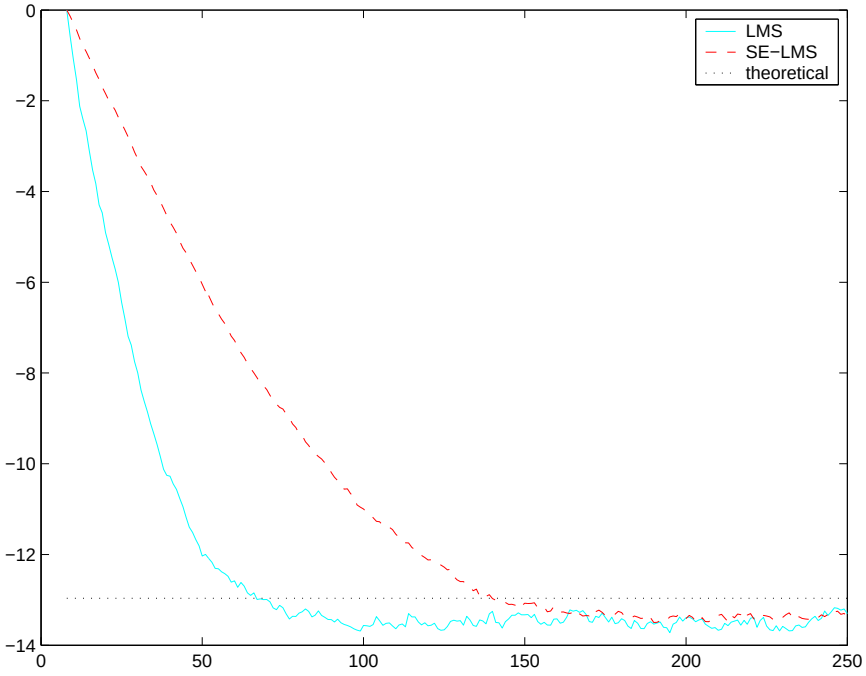


Figure 8: SE-LMS versus LMS.

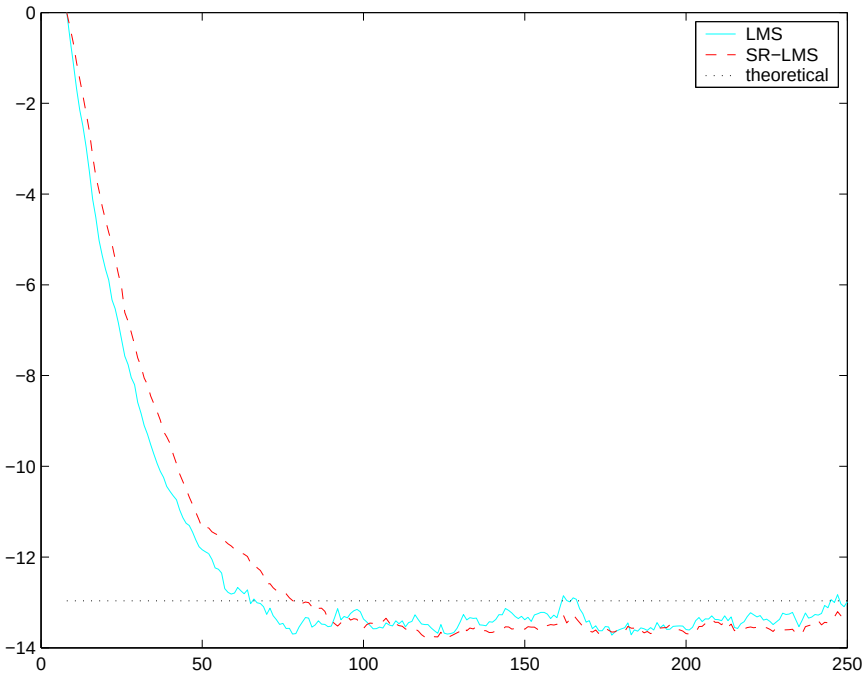


Figure 9: SR-LMS versus LMS.

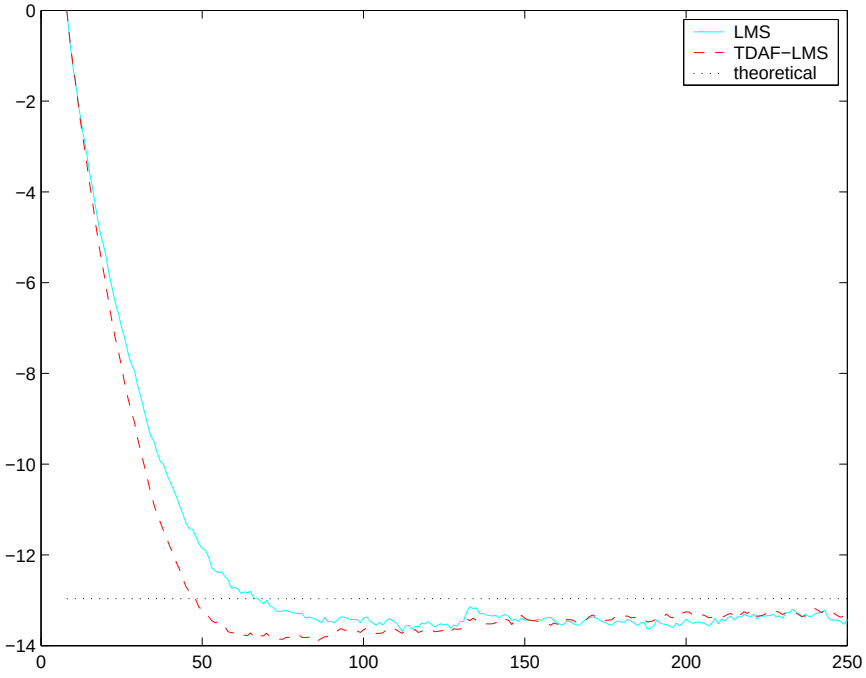


Figure 10: DFT-based TDAF versus LMS ($\hat{\boldsymbol{\lambda}}(0) = \mathbf{1}$).

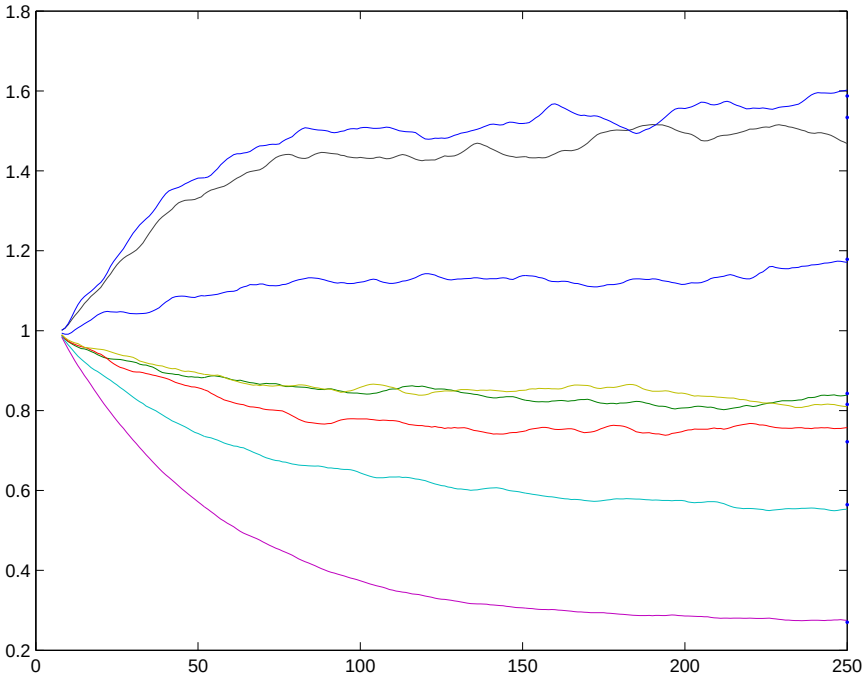


Figure 11: Evolution of $\hat{\boldsymbol{\lambda}}(n)$ (from $\hat{\boldsymbol{\lambda}}(0) = \mathbf{1}$) using $\gamma = 0.02$.

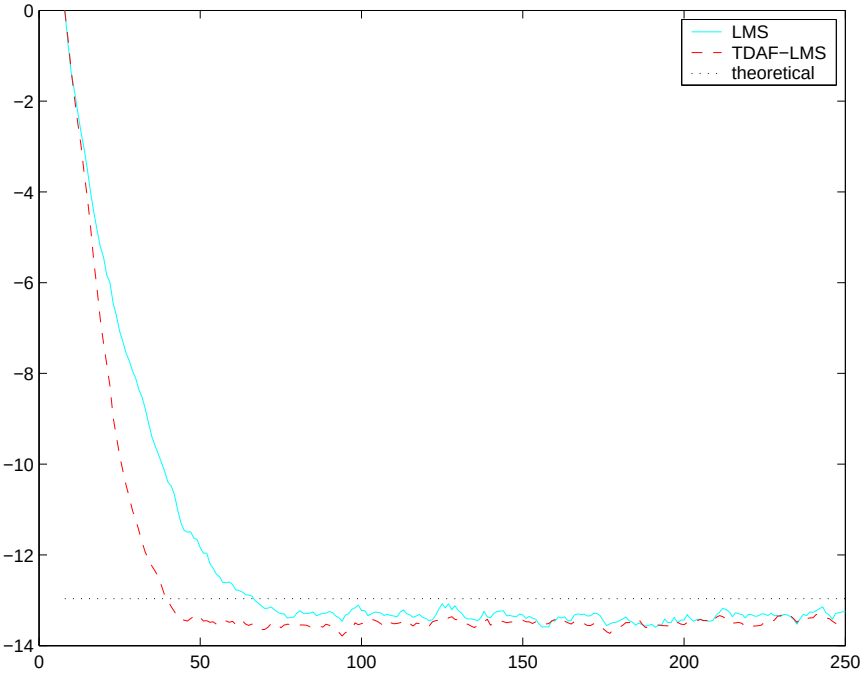


Figure 12: DFT-based TDAF versus LMS ($\hat{\boldsymbol{\lambda}}(0) = \boldsymbol{\lambda}$).

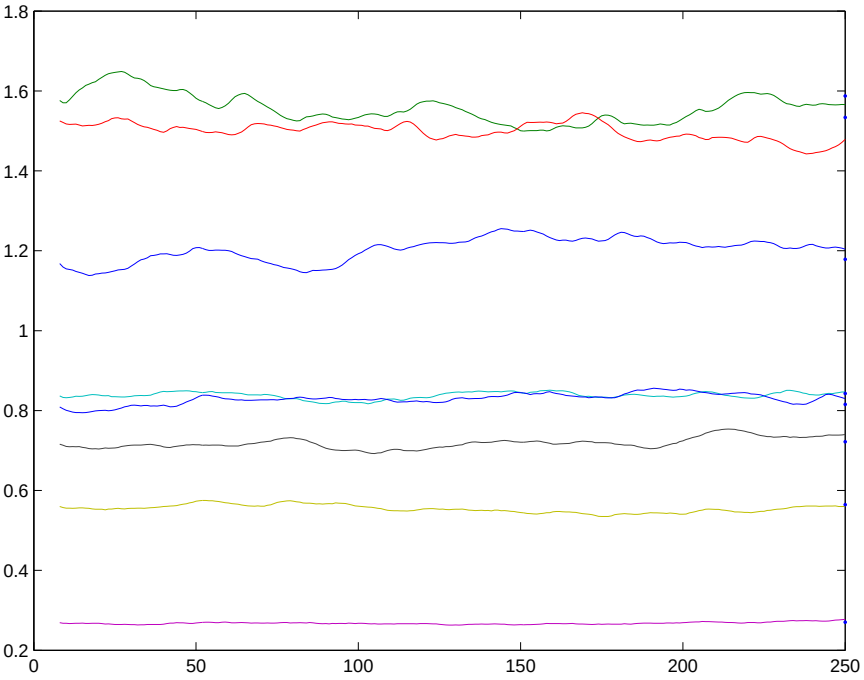


Figure 13: Evolution of $\hat{\boldsymbol{\lambda}}(n)$ (from $\hat{\boldsymbol{\lambda}}(0) = \boldsymbol{\lambda}$) using $\gamma = 0.02$.