Advanced Digital Signal Processing

Adaptive Linear Prediction Filter

(Using The RLS Algorithm)

Erick L. Oberstar

Adaptive Linear Prediction Filter Using the RLS Algorithm

A complete analysis/discussion of my results is given. Comparisons are made between my experimental results and theory.

The project is organized according to problem number, e.g., 1, 2a, 2b, etc. For each, a summary that highlights the results obtained and contains discussion comments is included.

This project is a continuation the previous Adaptive Linear Prediction Filter – a four tap linear predictor with an input consisting of
two sinusoids plus noise.

1. The input $\{u(n)\}$ is applied to an adaptive four tap predictor that uses the RLS algorithm. Initialize the algorithm with d = .004. Consider the same parameter sets as given in part 2) of Project 2.

a) Use the computer to generate a 300 sample sequence representing $\{u(n)\}$ for parameter set(i). Average over 200 ensembles to plot the learning curves of the RLS algorithm for $\lambda = .999$ and $\lambda = .98$. Estimate the corresponding values of the misadjustment by time averaging over the last 200 iterations of the ensemble averaged learning curves.

b) For parameter set (i), also estimate mean values for the tap weights that result from the RLS algorithm for $\lambda = .999$ and $\lambda = .98$. You may do this by averaging the steady state values of the tap weights (obtained at the last iteration) over 200 ensembles.

c) Repeat parts a) and b) for the parameter set (iii).

2.

a) Plot the theoretical autoregressive power spectrum estimate for parameter sets (i), (ii), and (iii). This is given by the inverse of the magnitude squared of the prediction error filter frequency response.

$$S_{AR}(\omega) = \frac{1}{\left|A(e^{j\omega})\right|^2}$$

where $A(z)$ is the prediction error filter transfer function.

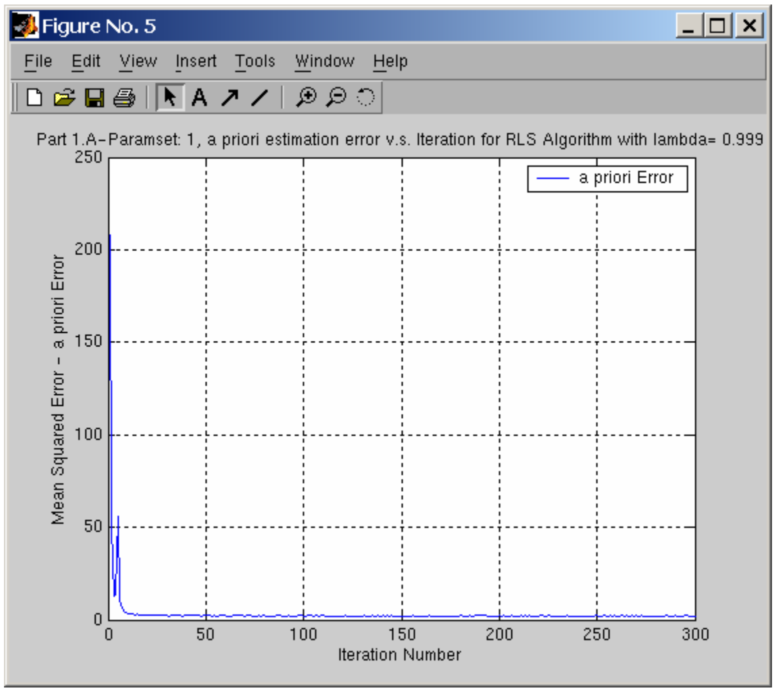
b) Plot the autoregressive spectrum estimates for 10 trials (plot all 10 on the same graph) using the LMS predictor coefficients obtained with $\mu = 10^{-6}$ for parameter set (iii) after 5000 iterations. Use the coefficients at the last iteration.

c) Repeat b) for the RLS algorithm using l = .999 (only evaluate 1000 iterations).

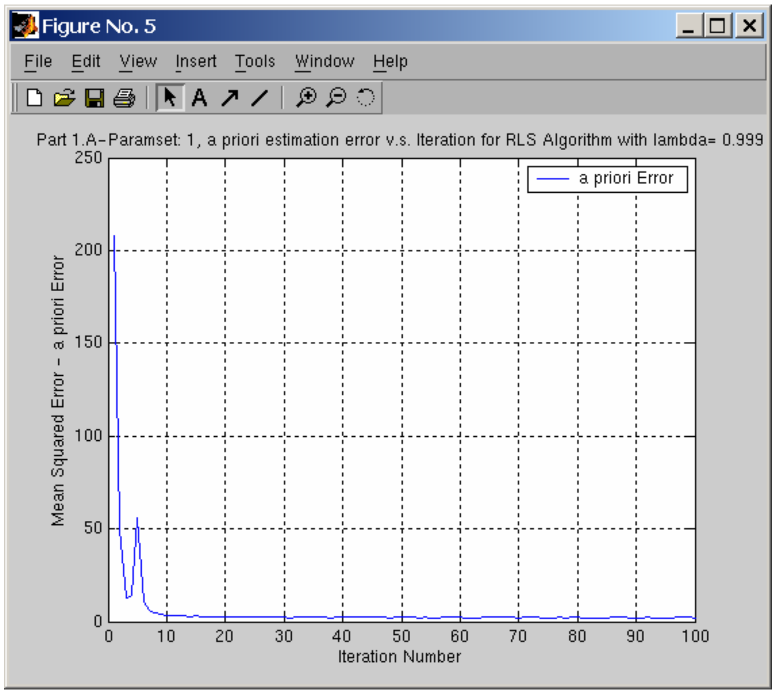
Compare, contrast, and discuss your results. Also discuss the differences you observe between the RLS algorithm and LMS algorithm (Project 2).

PART 1.a.i

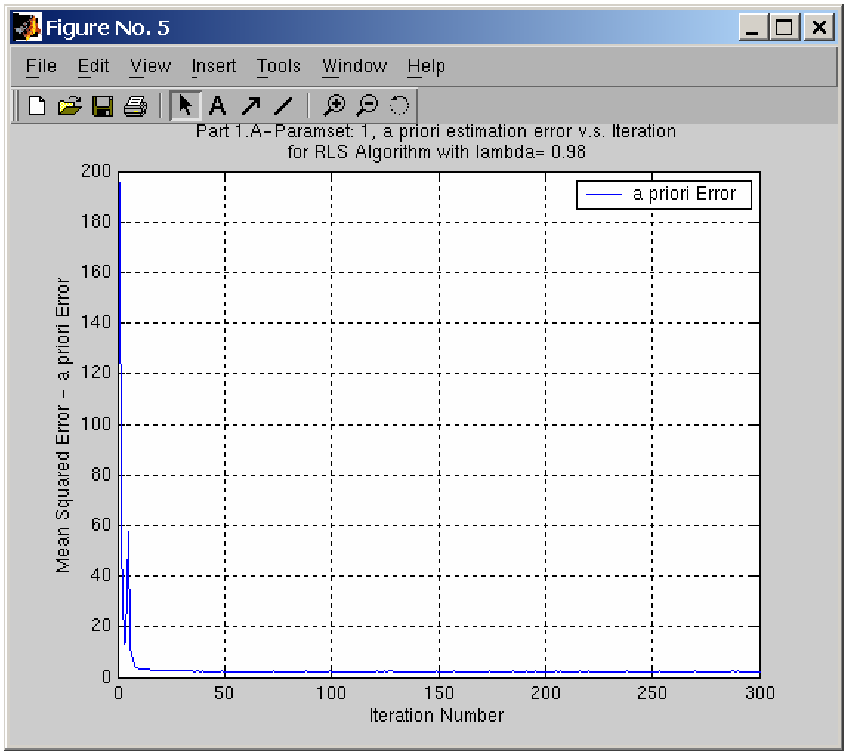
The RLS algorithm was implemented and applied to 300 statistically independent input sequences each 500 points long for forgetting factors $\lambda = .999$ and $\lambda = .98$ for parameter set 1 ($A^2 = 100$, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$). A plot of the ensemble averaged *a priori* error $\xi(n)$ was generated to show convergence of the algorithm. The plot showing the learning curve for $\lambda = .999$ is shown below:



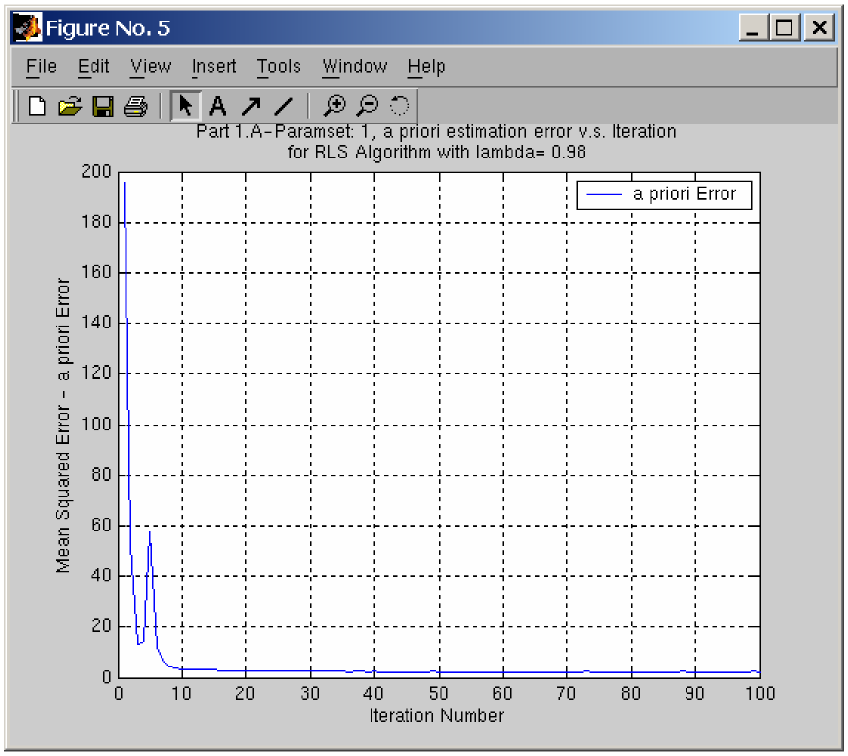
The above plot zoomed in is shown below:



The plot showing the learning curve for $\lambda = .98$ is shown below:



The above plot zoomed in is shown below:

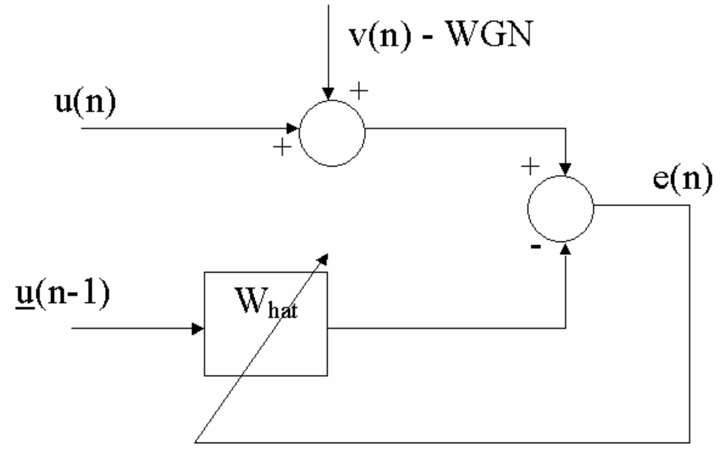


The misadjustment for parameter set 1 (i) was calculated by averaging the last 200 iterations of the *a priori* estimation error $\xi(n)$ for each forgetting factor $\lambda = 0.999$ and $\lambda = 0.98$. The experimental misadjustment for $\lambda = 0.999$ was calculated by Matlab to be $J_{ex}/J_{min} = 0.0123123$ (from iterations 300 – 500). The experimental misadjustment for $\lambda = 0.98$ was calculated by Matlab to be $J_{ex}/J_{min} = 0.05342552$ (from iterations 300 – 500). The theoretical excess MSE for the RLS algorithm is zero. This seems to agree with the experimentally calculated value. It is however off.

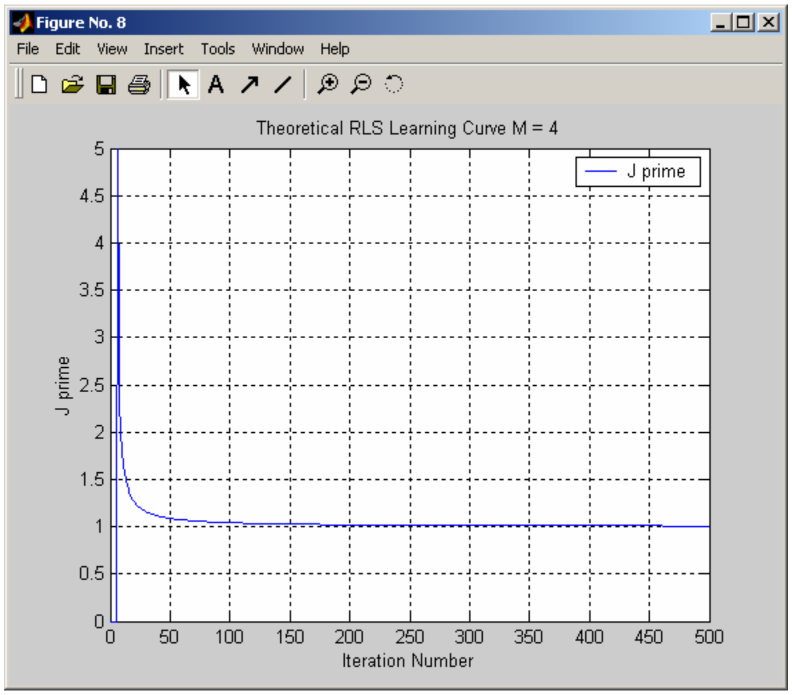
Explanation for misadjustment error:
As the number of iterations n approaches ∞ the mean-squared error $J'(n) \rightarrow \sigma^2$. The misadjustment property of the RLS algorithm assumes a forgetting factor $\lambda = 1$ i.e. infinite memory. Our experimental results were derived form λ's $\neq 1$ i.e. not infinite memory.

It is important to note that the RLS algorithm has converged by iteration 8 which corresponds to the expected convergence by iteration 2*M where M is the number of taps.

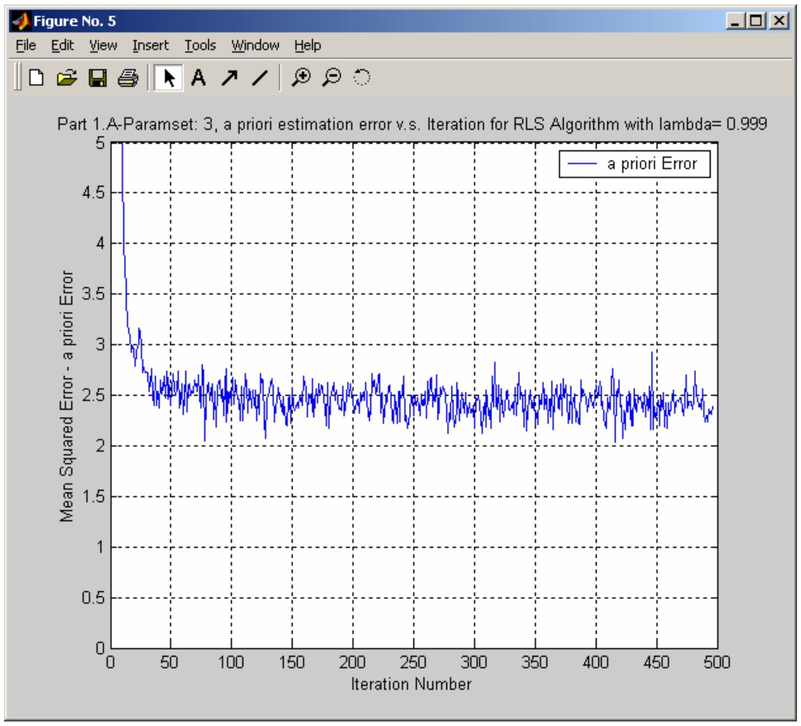
The theoretical RLS learning curve given by $J'(n) = \sigma^2 + \dfrac{M\sigma^2}{n - M - 1}$, for n>M+1 & $\lambda = 1$ and σ^2 is the variance of the measurement error $e_0(n)$, was plotted to see the theoretical rate of convergence and theoretical J_{min}. The block diagram for our linear prediction system is shown below:



For our linear prediction system $e_0(n)$(in the textbook) is merely the noise induced into the signal (v(n)). The *a priori* estimation error is defined as $e(n) = (u(n) + v(n)) - w_{hat}(n)\underline{u}(n-1)$. Once the algorithm has converged we are doing a good job estimating u(n) but not the noise contribution. Therefore J'_{min} is merely the noise power $\sigma^2 = 1$. The plot of the theoretical learning curve is shown below:



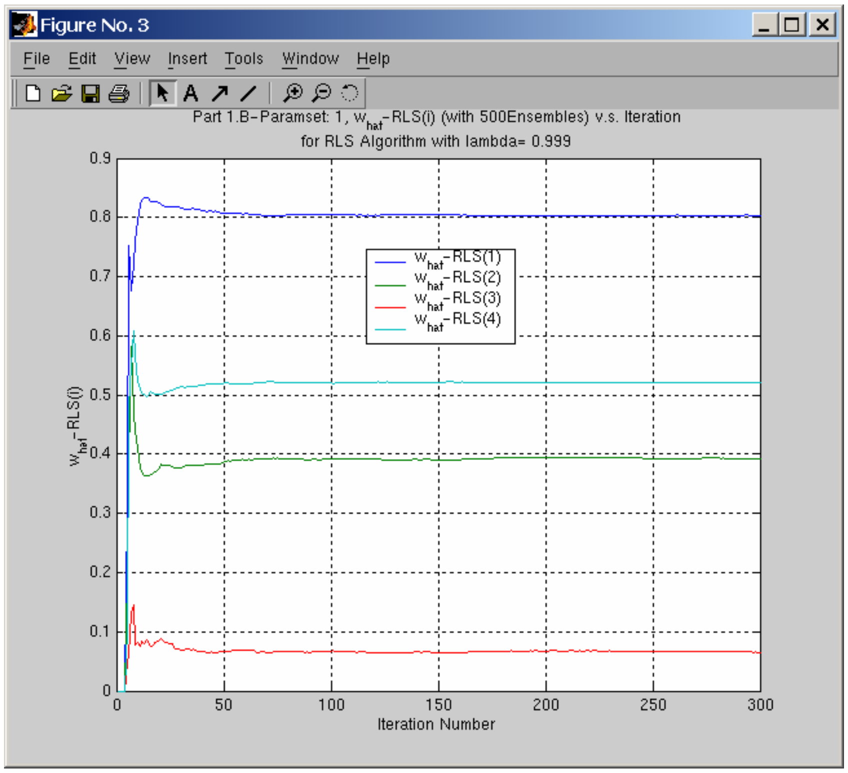
When comparing this to a zoomed plot for λ=0.999 the impact of the forgetting ≠ 1. The misadjustment does not equal zero. This zoomed in plot showing the misadjustment at a level higher than the theoretical learning curve. The plot showing non-zero misadjustment is shown below:



It is important to note that the theoretical learning curve agrees with the experimental curve convergence rate. However the experimental curve shows some misadjustment since the forgetting factor was less than one. I.e. the filter does not have infinite memory. Another important point to note is that difference between λ = .999 and λ = .98 plots is extra misadjustment for λ = .98. This results from less "memory" and agrees with theory.

PART 1.b.i

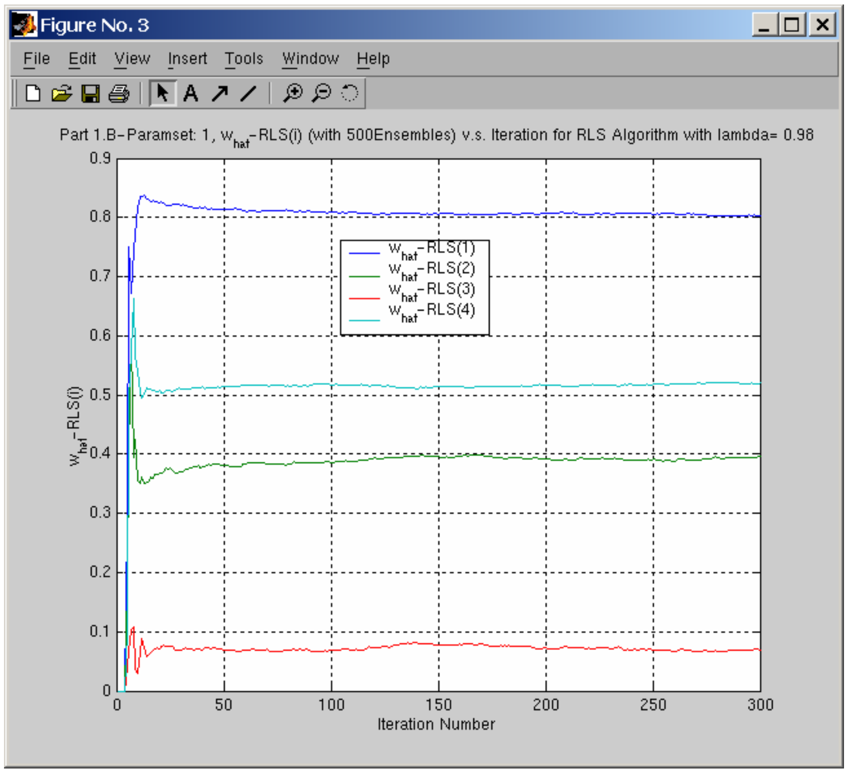
The ensemble averaged tap weights resulting from the RLS algorithm were plotted for both forgetting factors λ = .999 and λ = .98 for parameter set 1 (A^2 = 100, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$). The plot of tap weight learning curve for λ = .999 is shown below:



For λ = .999 the estimated mean tap weights were calculated by averaging the steady state values of the tap weights (obtained at the last iteration) over 200 ensembles for parameter set 1 (A^2 = 100, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$). The optimum tap weights were also calculated for comparison. The mean tap weights agree with the optimum solution. Note that the estimated mean tap weights are converged shortly after 2M iterations. This agrees with the theory.

(A^2 = 100, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$)	Optimum Tap Weights	Mean Tap Weights for λ = .999 (w_hat)
w_hat(1)	0.56631028353458 + 0.56631028353460i	0.56503085650269 - 0.56496139783470i
w_hat(2)	-0.00000000000000 + 0.39679271355726i	-0.00008190402483 - 0.39712912409408i
w_hat(3)	0.04787561926788 - 0.04787561926788i	0.04708257950345 + 0.04703631972970i
w_hat(4)	0.52189773566908 - 0.00000000000000i	0.52217914849469 - 0.00001301307500i

The plot of tap weight learning curve for λ = .98 is shown below:

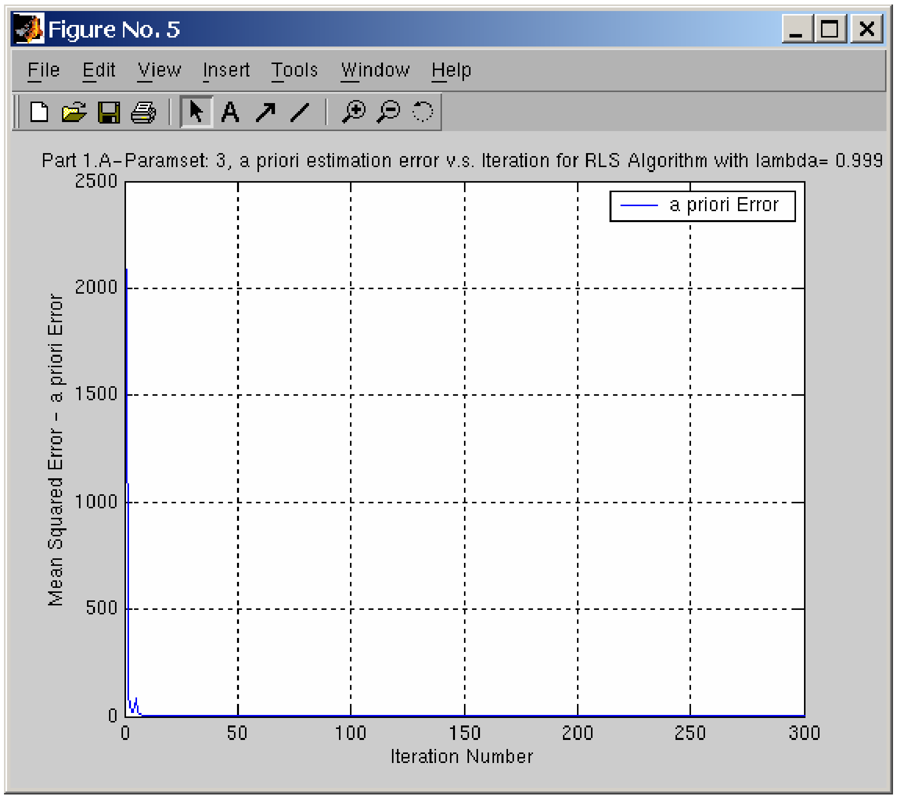


For λ = .98 the estimated mean tap weights were calculated by averaging the steady state values of the tap weights (obtained at the last iteration) over 200 ensembles for parameter set for parameter set 1 ($A^2 = 100$, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$). The optimum tap weights were also calculated for comparison. The mean tap weights agree with the optimum solution. Note that the estimated mean tap weights are converged shortly after 2M iterations. This agrees with the theory.

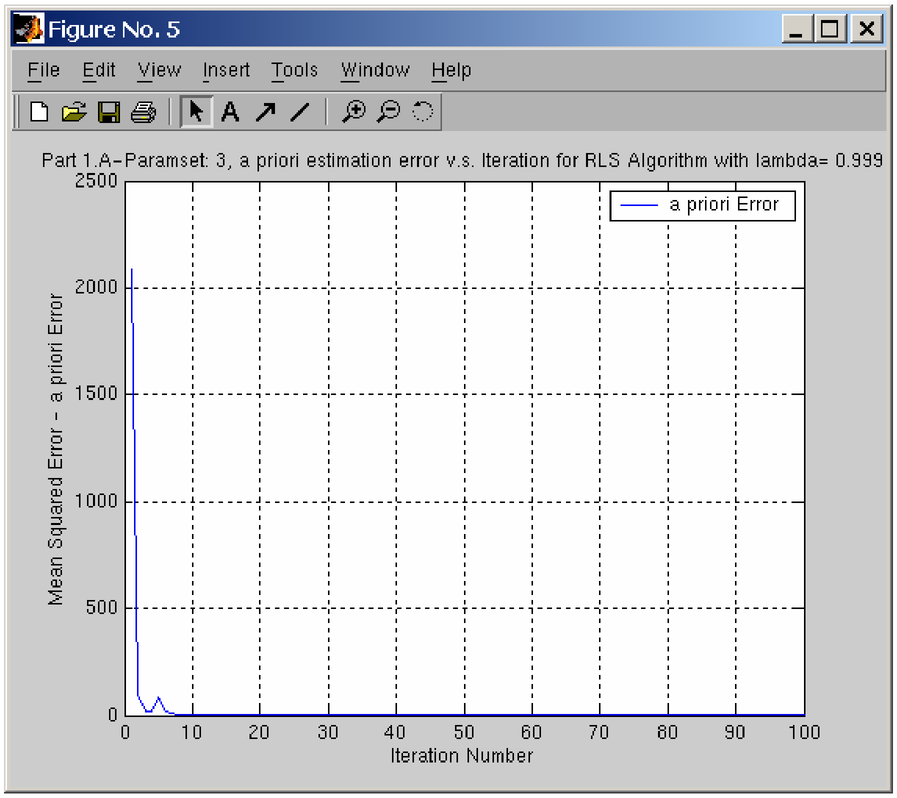
($A^2 = 100$, $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$)	Optimum Tap Weights	Mean Tap Weights for λ = .98 (w_hat)
(1)	0.56631028353458 + 0.56631028353460i	0.56494781187821 - 0.56510290392408i
(2)	-0.00000000000000 + 0.39679271355726i	-0.00012860713464 - 0.39728953709510i
(3)	0.04787561926788 - 0.04787561926788i	0.04716739116828 + 0.04718307413645i
(4)	0.52189773566908 - 0.00000000000000i	0.52211861707578 + 0.00005971141632i

PART 1.c.a

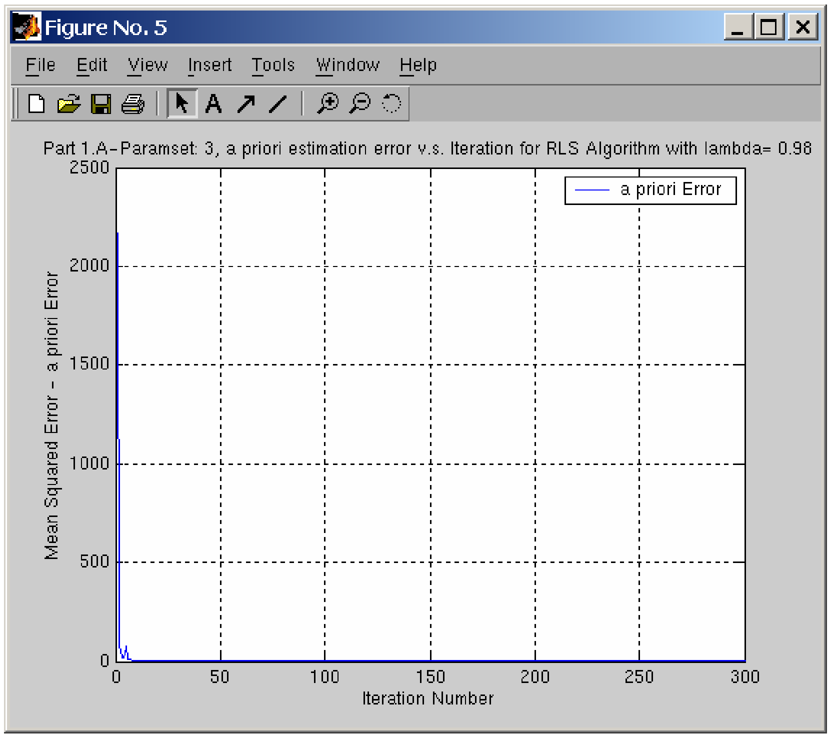
The ensemble averaged tap weights resulting from the RLS algorithm were plotted for both forgetting factors λ = .999 and λ = .98 for parameter set 3 ($A^2 = 100$, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$). The plot of the learning curve for λ = .999 is shown below:



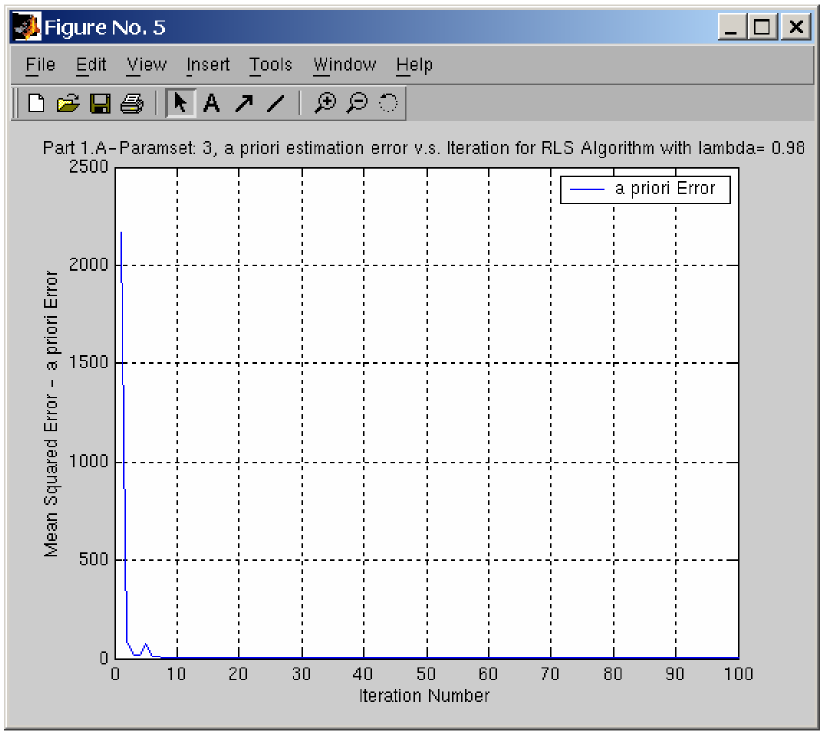
The above plot zoomed in is shown below:



The plot showing the learning curve for λ = .98 is shown below:



The above plot zoomed in is shown below:



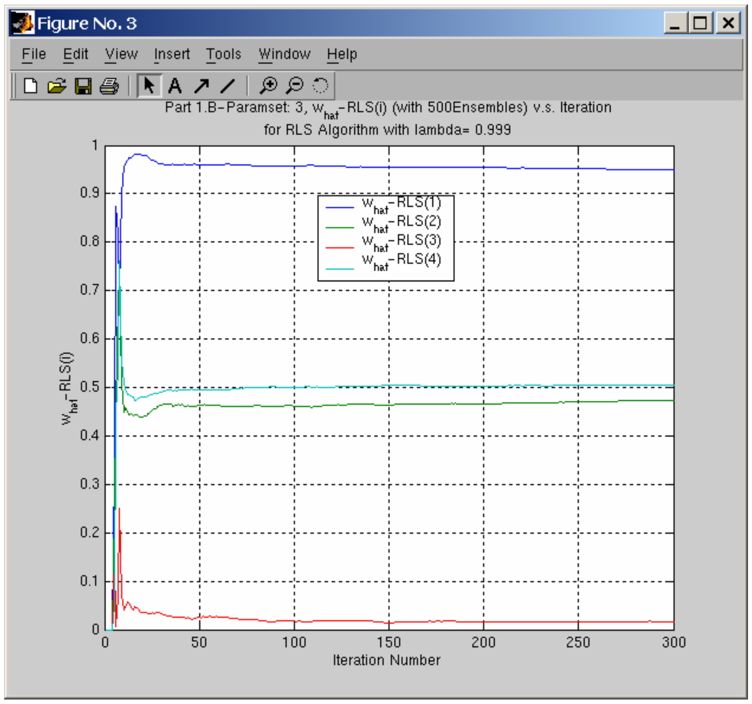
The misadjustment for parameter set 3 (iii) was calculated by averaging the last 200 iterations of the *a priori* estimation error $\xi(n)$ for each forgetting factor $\lambda = 0.999$ and $\lambda = 0.98$. The experimental misadjustment for $\lambda = 0.999$ was calculated by Matlab to be $J_{ex}/J_{min} = 0.01279322$ (from iterations 300 – 500). The experimental misadjustment for $\lambda = 0.98$ was calculated by Matlab to be $J_{ex}/J_{min} = 0.0592008$ (from iterations 300 – 500). The theoretical excess MSE for the RLS algorithm is zero. This seems to agree with the experimentally calculated value. It is however off.

Explanation for misadjustment error:
As the number of iterations n approaches ∞ the mean-squared error $J'(n) \rightarrow \sigma^2$. The misadjustment property of the RLS algorithm assumes a forgetting factor $\lambda = 1$ i.e. infinite memory. Our experimental results were derived form λ's $\neq 1$ i.e. not infinite memory.

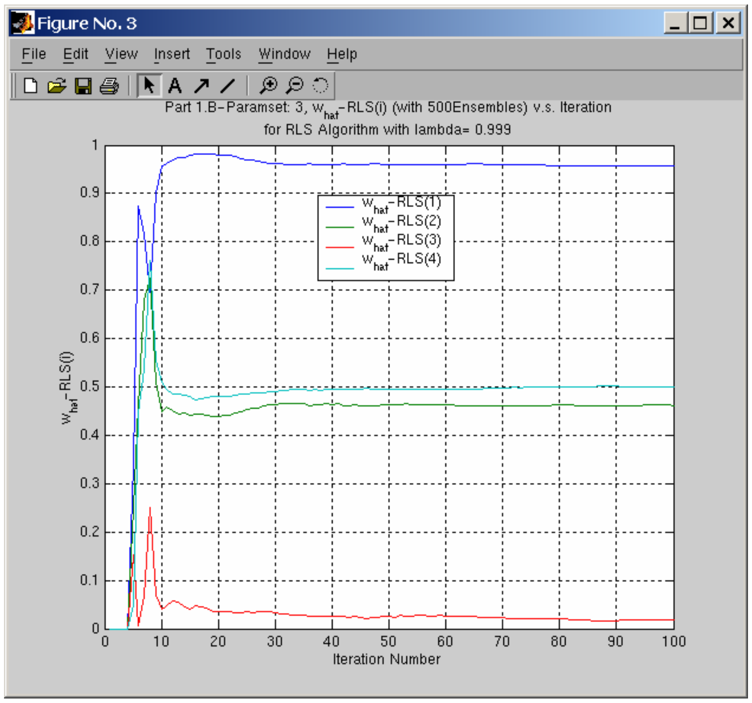
It is important to note that the RLS algorithm has converged by iteration 8 which corresponds to the expected convergence by iteration 2*M where M is the number of taps.

PART 1.c.b

The ensemble averaged tap weights resulting from the RLS algorithm were plotted for both forgetting factors $\lambda = .999$ and $\lambda = .98$ for parameter set 3(iii) ($A^2 = 100$, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$). The plot of tap weight learning curve for $\lambda = .999$ is shown below:



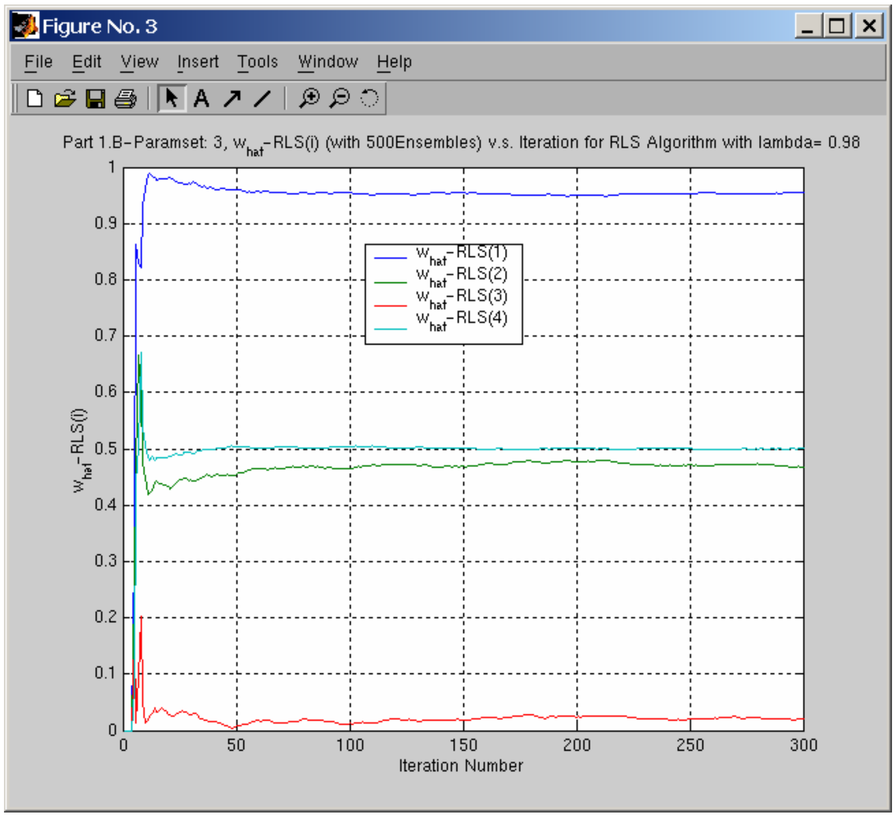
The above plot zoomed in is shown below:



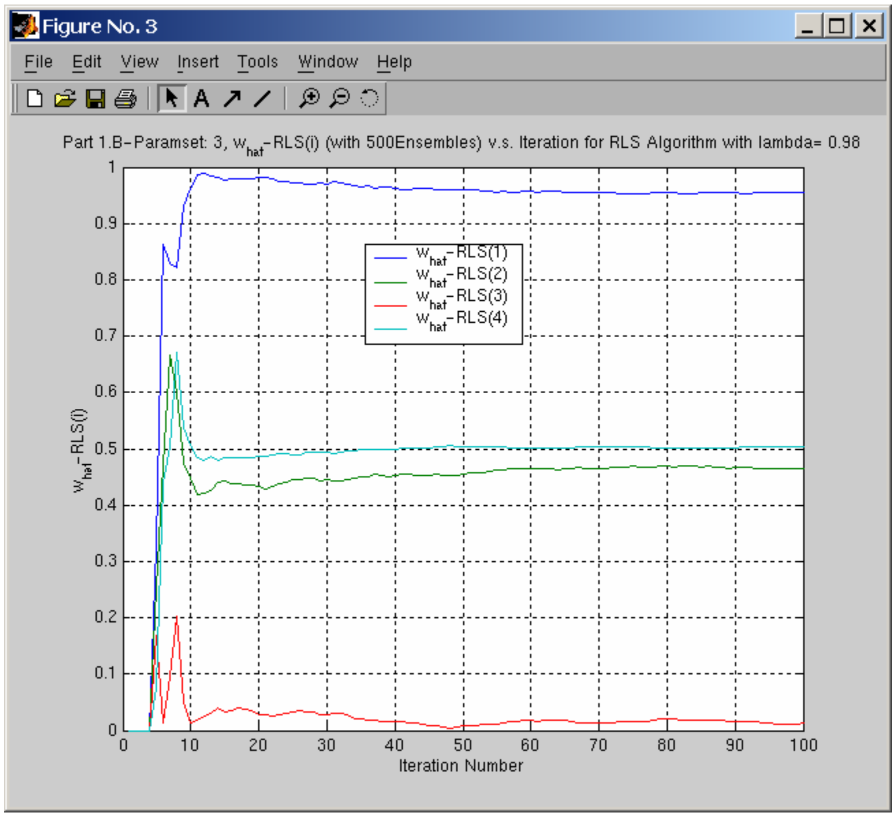
For λ = .999 the estimated mean tap weights were calculated by averaging the steady state values of the tap weights (obtained at the last iteration) over 200 ensembles for parameter set 3 (iii) (A^2 = 100, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$). The optimum tap weights were also calculated for comparison. The mean tap weights agree with the optimum solution. Note that the estimated mean tap weights are converged shortly after 2M iterations. This agrees with the theory.

(A^2 = 1000, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$)	Optimum Tap Weights	Mean Tap Weights for λ = .999 (w_hat)
w_hat(1)	0.78845661831349 + 0.52682986928323i	0.76467893489554 - 0.51087198983165i
w_hat(2)	0.18202160980670 + 0.43943903903978i	0.19724786692594 - 0.47557816648569i
w_hat(3)	0.00297643857249 - 0.01496356718147i	0.00286372271405 + 0.01364763321467i
w_hat(4)	0.35749375919968 - 0.35749375919975i	0.36705378917505 + 0.36685882241925i

The plot of tap weight learning curve for λ = .98 is shown below:



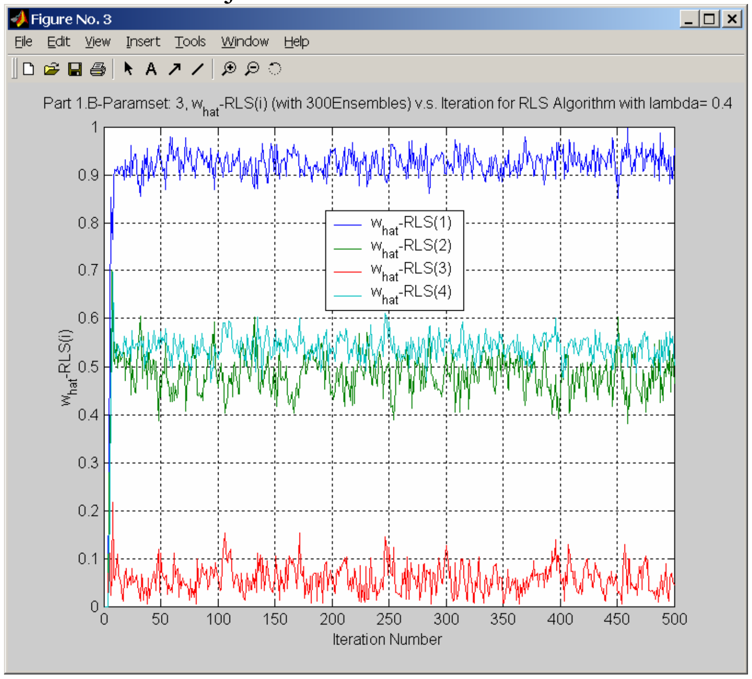
The above plot zoomed in is shown below:



For λ = .98 the estimated mean tap weights were calculated by averaging the steady state values of the tap weights (obtained at the last iteration) over 200 ensembles for parameter set 3 (iii) (A^2 = 100, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$). The optimum tap weights were also calculated for comparison. The mean tap weights agree with the optimum solution. Note that the estimated mean tap weights are converged shortly after 2M iterations. This agrees with the theory.

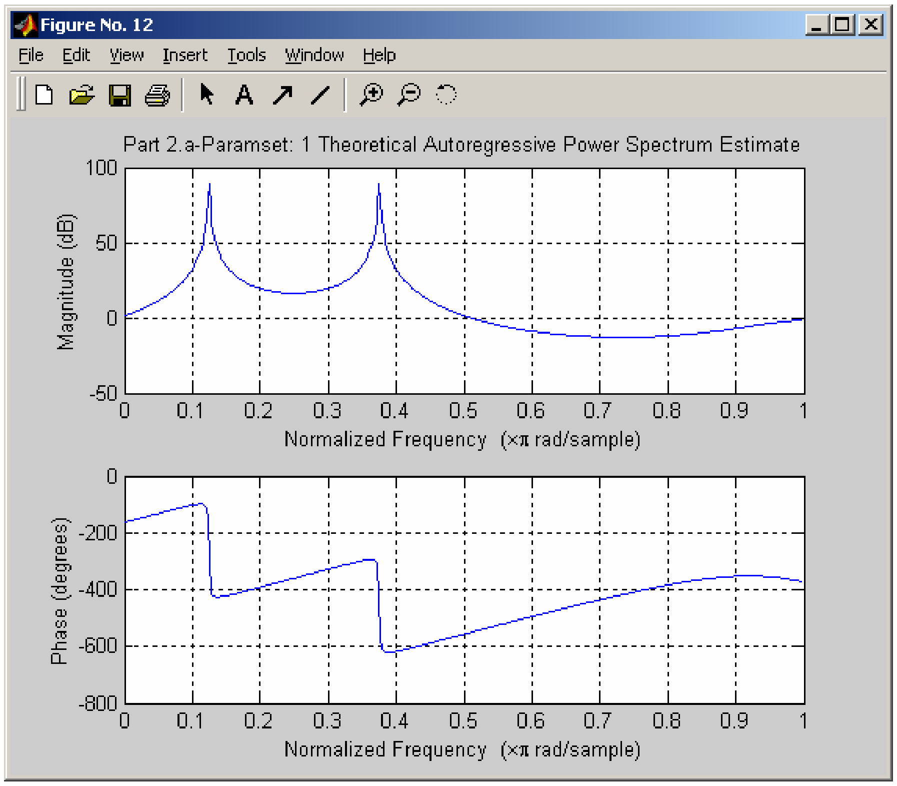
(A^2 = 1000, $\omega_1 = \pi/8$, $\omega_2 = \pi/4$)	Optimum Tap Weights	Mean Tap Weights for λ = .98 (w_hat)
(1)	0.78845661831349 + 0.52682986928323i	0.76412729171377 - 0.51028044279843i
(2)	0.18202160980670 + 0.43943903903978i	0.19705278596037 - 0.47559943653436i
(3)	0.00297643857249 - 0.01496356718147i	0.00355593605463 + 0.01401276330474i
(4)	0.35749375919968 - 0.35749375919975i	0.36755876463973 + 0.36747390075505i

As an exercise in curiosity I plotted the ensemble average tap weights for λ = .4. It is interesting to note the obvious increase in misadjustment.



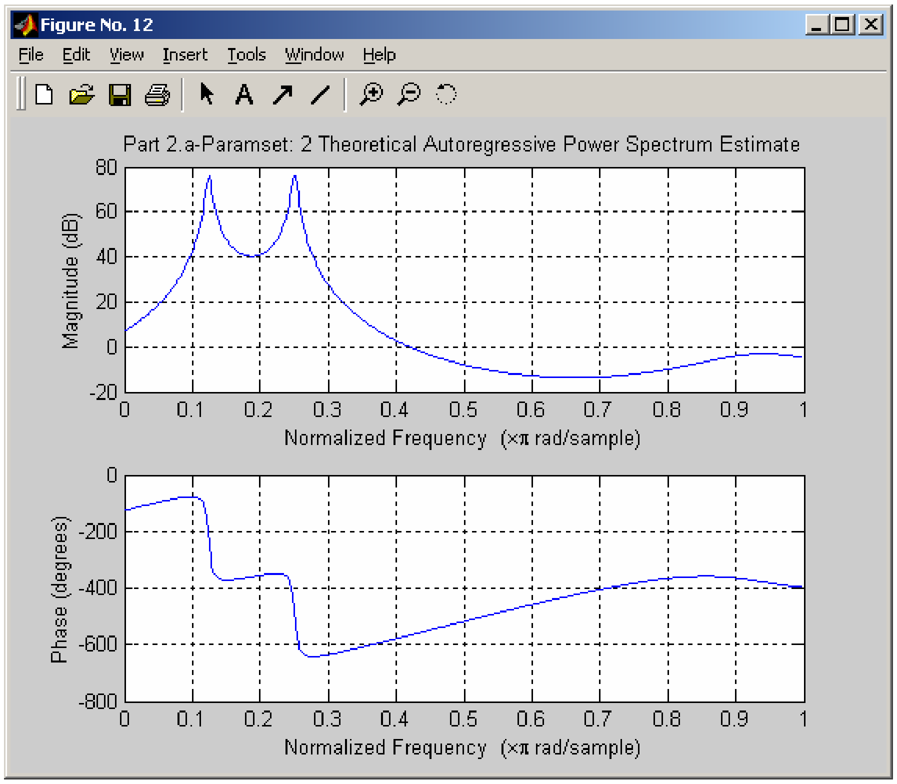
PART 2.a

The theoretical autoregressive power spectrum estimate for parameter sets 1 (i), 2 (ii), and 3 (iii) were plotted using the theoretical optimum tap weights for each parameter set. The theoretical autoregressive power spectrum estimate for parameter set 1 (i) is shown below:



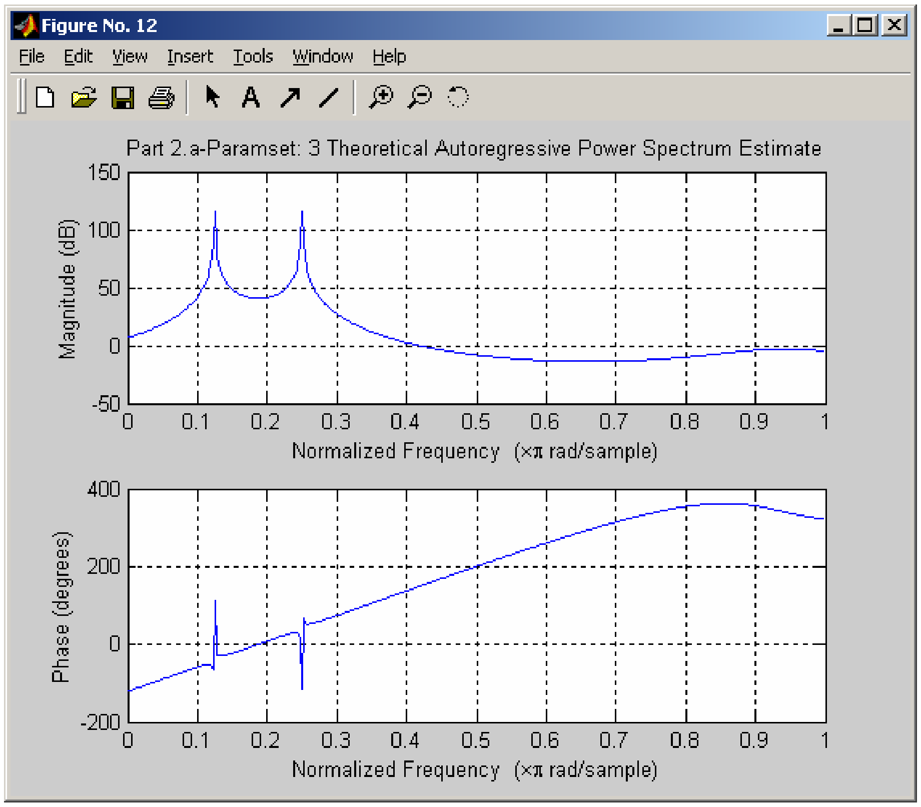
The two peaks correspond to the sinusoids at $\omega_1 = \pi/8$, $\omega_2 = 3\pi/8$.

The theoretical autoregressive power spectrum estimate for parameter set 2 (ii) is shown below:



The two peaks correspond to the sinusoids at $\omega_1 = \pi/8$, $\omega_2 = \pi/4$. Note that the amplitude of the peaks for parameter set 1 and 2 are the same (since they have the same power)

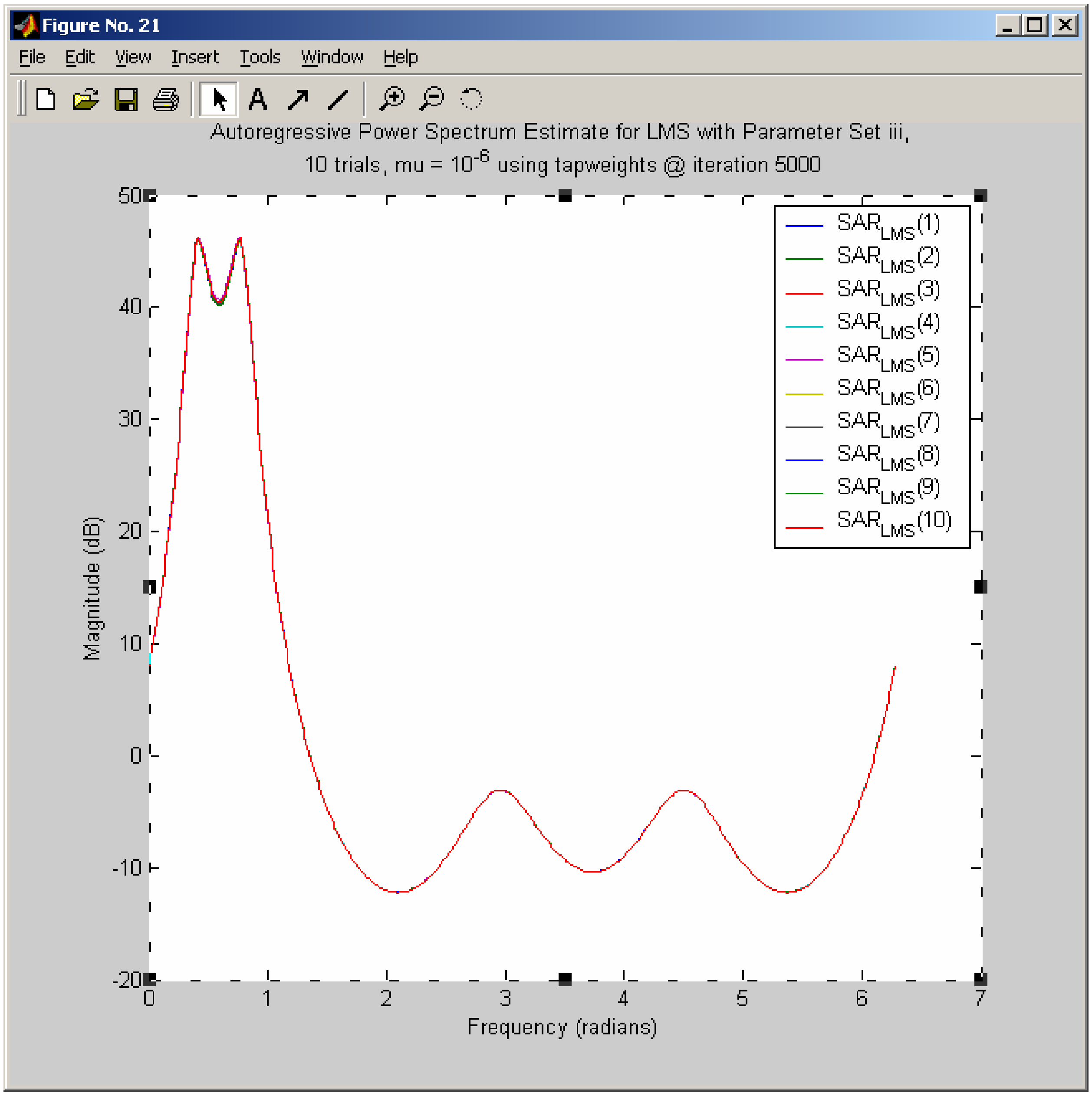
The theoretical autoregressive power spectrum estimate for parameter set 3 (iii) is shown below:



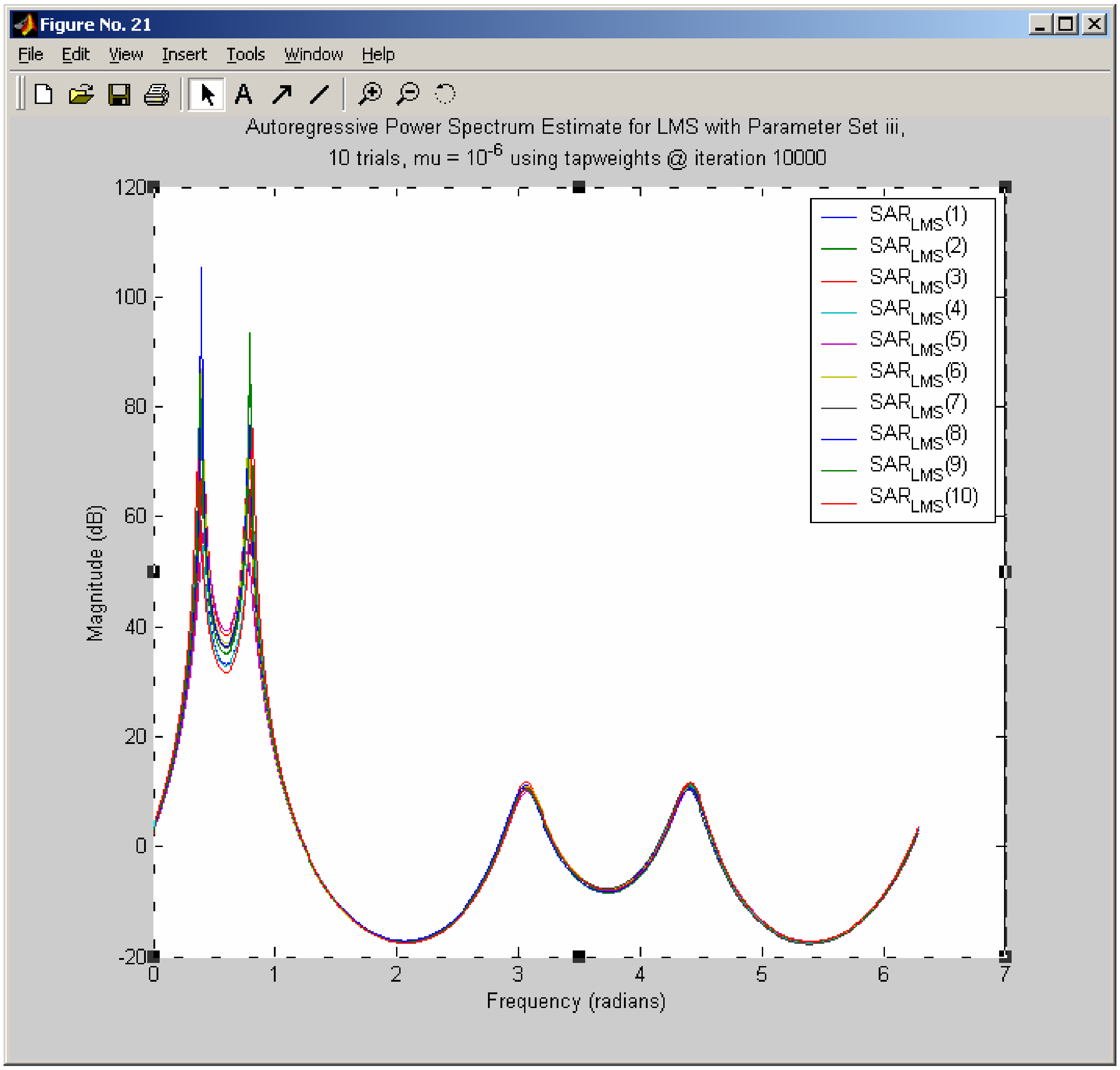
The two peaks correspond to the sinusoids at $\omega_1 = \pi/8$, $\omega_2 = \pi/4$. Note that the sinusoids for parameter set 3 are at the same frequencies as parameter set 2. However note amplitude is larger for parameter set 3 as expected.

PART 2.b

The autoregressive power spectrum estimates of ten trials (ensembles) were generated using the LMS predictor coefficients obtained (at the last iteration) with a step size of $\mu = 10^{-6}$ for parameter set 3 (iii) after 5000 iterations. This plot is shown below



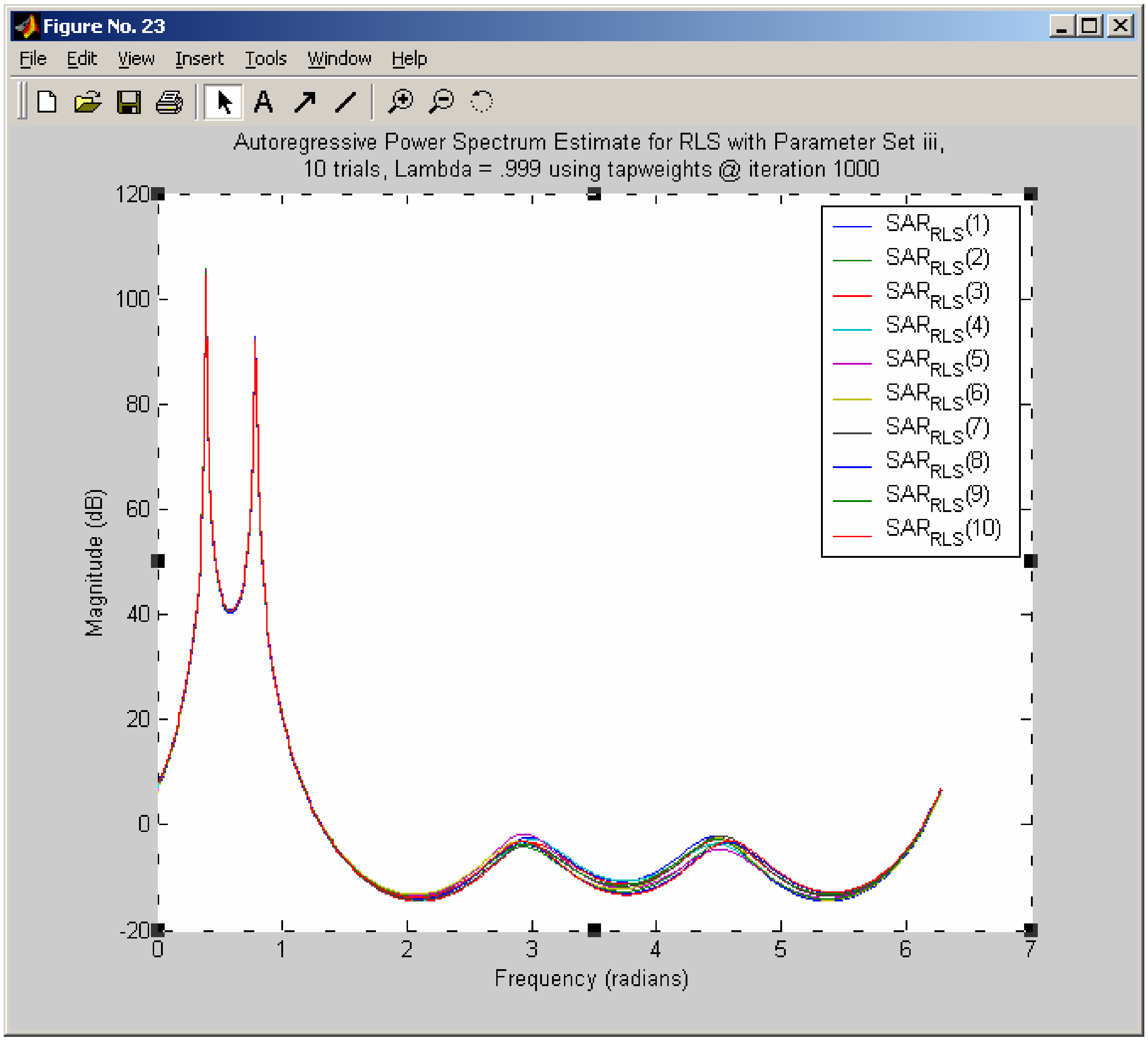
Note the peaks appear at the correct frequencies $\omega_1 = \pi/8 = 0.395$, $\omega_2 = \pi/4 = 0.785$) but the amplitude does not correspond to the theoretical plot for parameter set 3 (iii). This is because the LMS algorithm has not converged by iteration 5000. The autoregressive power spectrum estimate using LMS predictor coefficients was plotted again after allowing the LMS algorithm to converge (10000 iterations). This plot is shown below:



The above plot is in better agreement with the theoretical spectrum estimate since the tap weights have been allowed to converge.

PART 2.c

The autoregressive power spectrum estimates of ten trials (ensembles) were generated using the RLSS predictor coefficients obtained (at the last iteration) with a step size of $\lambda = .999$ for parameter set 3 (iii) after only 1000 iterations. This plot is shown below



This plot agrees well with the theoretical plot for parameter set 3 (iii).

Conclusion:
Note: The Matlab source code used to generate the plots follows this conclusion.

The RLS algorithm implemented in this project agrees well with the theoretical best-case tap weights and MSE/misadjustment. Obvious trade offs were observed with regards to forgetting factor versus misadjustment. The tap weights and *a priori* error converge quickly. For RLS the error curves have very small misadjustment. Although the misadjustment is not exactly zero it does get close. Since we do not have a forgetting factor $\lambda=1$, this is expected. The overall agreement of the experimental results with the theoretical results is good. The RLS algorithm has obvious benefits over the LMS algorithm. The most obvious is convergence rate, which in this experiment was a little more than an order of magnitude better than the LMS algorithm. The misadjustment of the RLS algorithm is better than that of the LMS algorithm. Since RLS is an order M^2 algorithm and LMS is only an order M algorithm, this performance increase comes at the cost of computational complexity. It is worth noting that the convergence of the RLS algorithm occurs at approximately the same iteration (2M) for all three parameter sets. Since the different parameter sets had different eigen value spreads, the experiment follows the theory that RLS rate of convergence is independent of eigen value spread for the ensemble averaged correlation matrix.

For this project we clearly violate the independence assumption of the statistical analysis of the RLS algorithm for the linear prediction filter. I.e. the linear prediction filter is subject to the shifting property of the input data. This same violation occurred in the LMS algorithm and it still converged. Despite this violation the RLS algorithm *a priori* estimation error and mean tap weight estimates still converge very close to zero misadjustment and the optimum tap weights. However the experimental misadjustment does not equal zero because the forgetting factor $\lambda=0.999$ and $\lambda=0.98$ was used rather than $\lambda=1$.

Other Conclusions:
Like for project 2 I still need a much faster computer with a boatload more RAM to run simulations in a more reasonable amount of time. I also still need a hobby to do while waiting for simulations to run on my old computer.