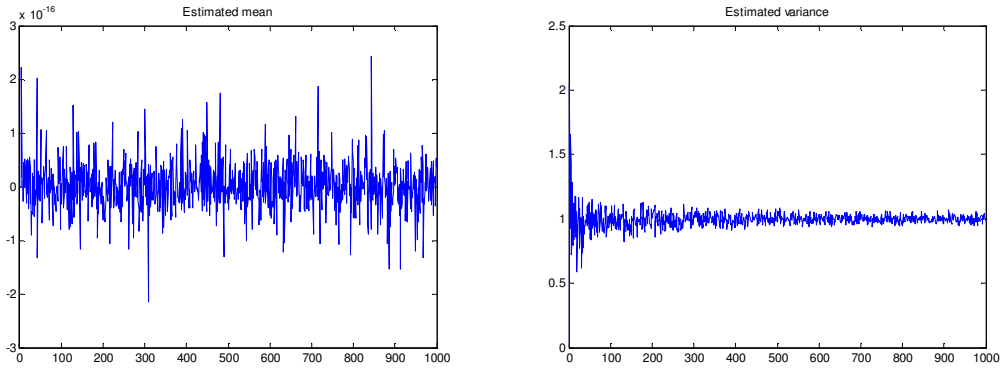
ECE 8110 – Homework Three

1. Estimated mean and variance of N points of zero-mean unit variance Gaussian signal.



The x-axis indicates the number of samples in the signal ranging from 1 to 1000. The mean quickly drives down to zero as the estimated mean is on the order of 10^-16 which converges to zero in a hurry. The variance is a slightly different story as for the first 100 sized samples it does not settle down to one. In fact, there is quite a bit of noise on the variance until the sample size begins to reach 500. This suggests that MLE tools can quickly return an accurate mean regardless of data size, but the variance takes a larger data set to return an accurate result.

2. Autocorrelation and Covariance.

For Gaussian autocorrelation: $x[n] = \mu + \Delta e$ with μ as the mean 0 and Δe as the variance

$$(\mu + \Delta e)(\mu + \Delta e) = \mu^2 + 2\mu\,\Delta e + \Delta e^2$$

With $\mu = 0$, the summation becomes the following

$$R(k) = \Sigma\, \Delta e_n\, \Delta e_{n-k}$$

For Gaussian covariance: $C_{ij} = \Sigma\, (\mu_{n-i} + \Delta e_{n-i})(\mu_{n-j} + \Delta e_{n-j})$

Again, $\mu_x = 0$ which results in the following

$$C_{ij} = \Sigma\, (\Delta e_{n-i})(\Delta e_{n-j})$$

The autocorrelation function is returning the error between each signal and the R(0) signal. The Fourier transform works to break down a signal into basic component parts that can be added together to reform/approximate the original signal. The autocorrelation of the Gaussian indicates how to scale the error from the R(0) term to replicate the other intervals of the signal. They both provide a map of how to rebuild a signal from inspect individual components of the signal. Thus the absolute value of any given autocorrelation, R(k), will always be a factor less than one of the of the R(0) signal.

The covariance matrix results in a mirrored full rank matrix that indexes the variance between the spacing of each sample subset. The matrix diagonal, where i=j, results as identity

when normalized against C_{11}. At least, I am rather sure this is how it should work out since the variance is set at unity so each diagonal should be unity, with the off diagonals indicting drift in reference to the other spacing options.

For sine autocorrelation: $R_x(a) = \Sigma[x(t)x(t-a)]$ where $x(t) = A\sin(ßt+\lambda)$

$A^2 \Sigma[\sin(ßt+\lambda)\sin(ß(t-a)+\lambda)]$ integrate!

$½pi \int A^2 \Sigma[\sin(ßt+\lambda)\sin(ß(t-a)+\lambda)] d\lambda$ from 0 to 2pi

Apply tri identity of $\sin(a)\sin(a+b) = ½[\cos(b)-\cos(2a+b)]$

$½pi \int A^2 \Sigma ½ [\cos(-2a) – \cos(4t+2\lambda-2a)] d\lambda$ from 0 to 2pi

$R_x(a) = ½ A^2\cos(-2a)$

For sine covariance: $C_{ij} = A\sin(2t-ai+\lambda)A\sin(2t-aj+\lambda)$

Again, tri identity from above; $u = 2t-ai+\lambda$ $v = 2t-aj+\lambda$

$\sin(u)\sin(v) = ½[\cos(u-v) – \cos(u+v)]$

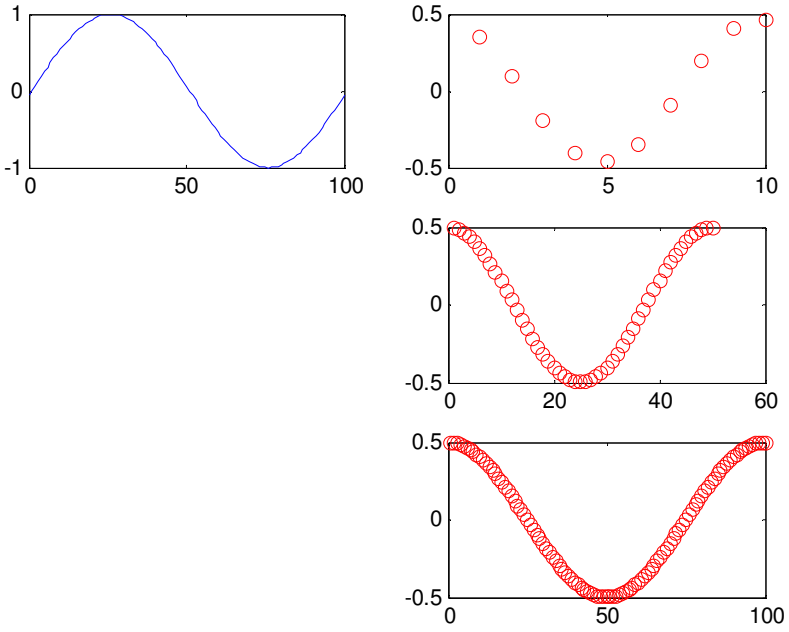
$A^2 /4pi \int \cos(2at + 2\lambda-ai+aj)d\lambda$ from 0 to 2pi

$U = 2at+2\lambda-ai-aj$ $du = 2d\lambda$

$A^2/2pi \cos(u)du$

$C_{ij} = A^2/2pi \cos(a(j-i))$

The rank of the sine covariance matrix should be full. The diagonal terms will be $A^2/2pi$ as $\cos(a(j-i))$ is always 1 when j=i. This makes sense as those terms are being correlated with themselves. Much in the same manner that R(0) returns the magnitude of the signal and is used to normalize the rest of the terms in autocorrelation.



The autocorrelation turns a sine wave into a cosine wave, as suggested by the above equations. The covariance matrix comes out as square and only really needs to be a 2x2.

C_ij =

0.0351 0.0466

0.0466 0.0618

3. Correlation.

To find the coefficient of the filter, 0.75, the best approach is to use the autocorrelation of the output signal. This only works because the equation of the filter is known, so we can mathematically solve for alpha.

$$Y'[n] = H[n]*X[n]; \qquad Y''[n] = X[n] - \alpha X[n-1] \qquad E = (Y'-Y'')^2 = 0$$

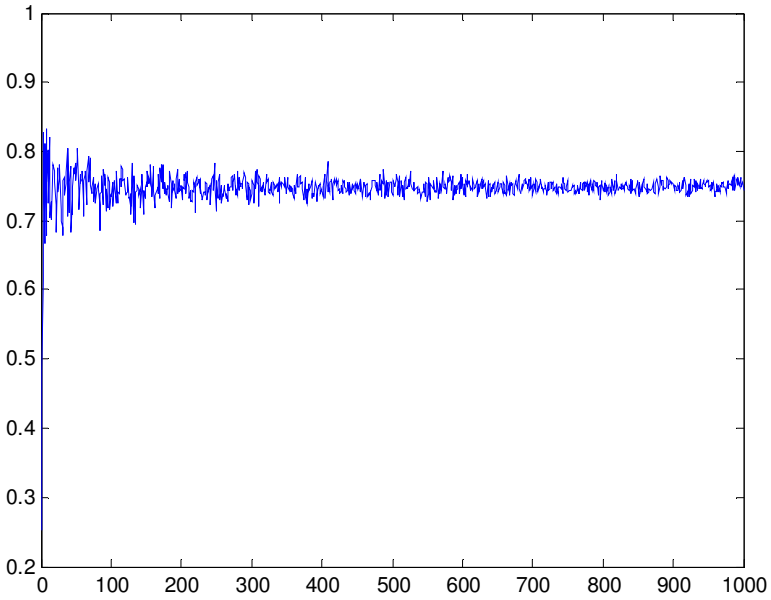
$$E = (Y[n] - \alpha Y[n-1])^2$$

$$dE/d\alpha = 0 = Y^2[n] - 2\alpha Y[n]Y[n-1] + \alpha^2 Y^2[n-1]$$

$$0 = -2Y[n]Y[n-1] + 2\alpha Y^2[n-1]$$

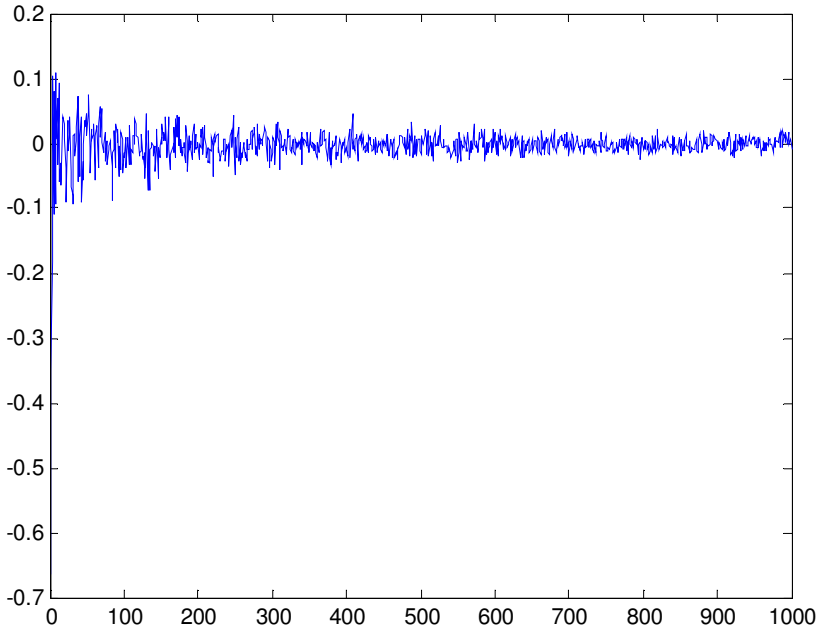
$$\alpha = Y[n]Y[n-1] / Y^2[n-1]$$

$$\alpha = R(1)/R(0)$$

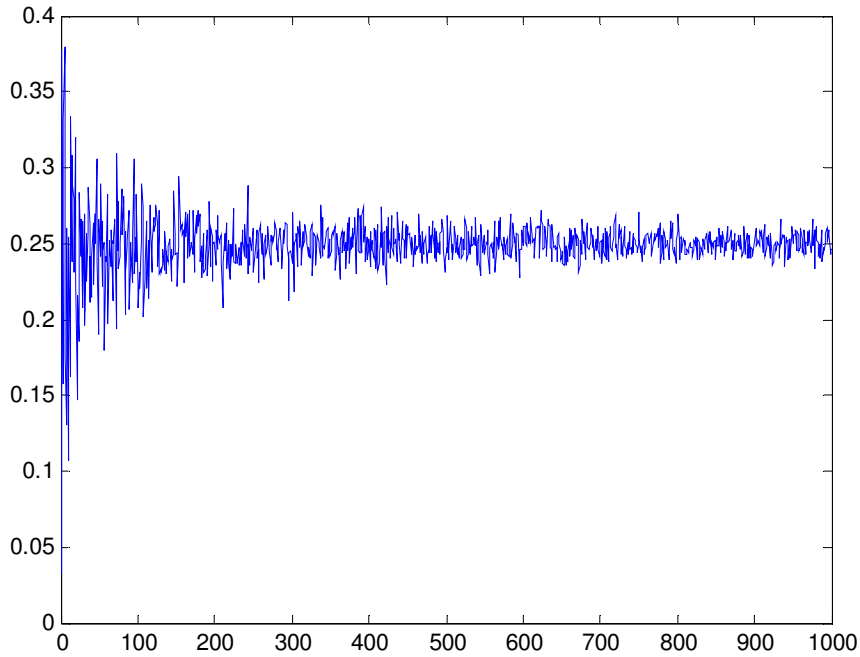


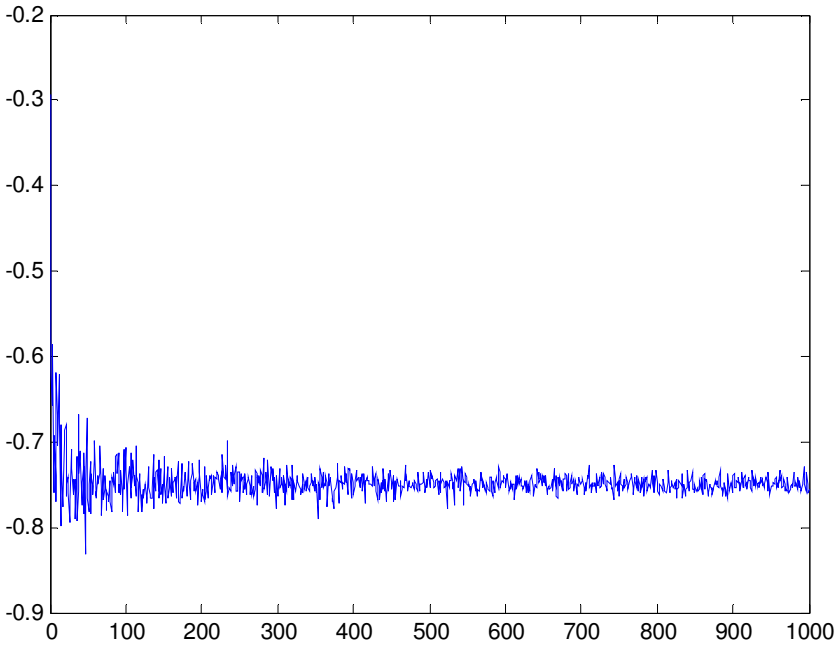
Test results indicate a great success with this approach. As the number of iterations increase (x-axis), alpha (y-axis) beings to approach the correct value. In this case the estimate of α quickly approaches the true value.

The error of this approach can be seen below, with x-axis representing interations and y-axis the error from the true value. The initial guess produces the most error, but the function quickly beings to converge as more data points are added to the algorithm.



This can be used for many filters, provided the form of the filter equation is known prior. Here is a case with $\alpha_1 = 0.25$ and $\alpha_2 = -0.75$. In both cases the true value of alpha is approach quite quickly, but noise does remain as the estimate is limited by the variations seen in the data. Adding a smoothing filter on the resultant would probably ensure convergence to a steady state solution.



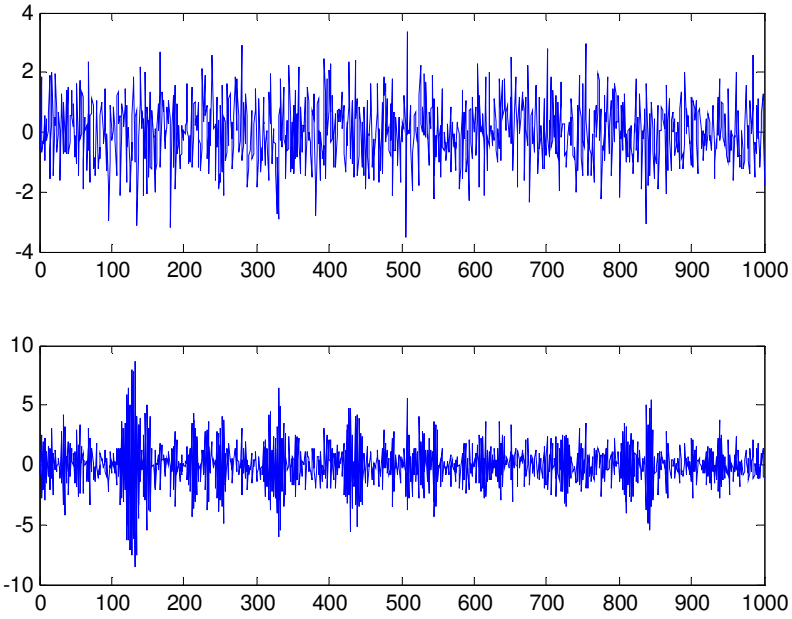


This style of estimation is a maximum likelihood estimate, because it takes the data itself and builds the model from the function that created the data. Without knowing the form of the filter, it would impossible to use ML as there is no equation to use as the model of the data.

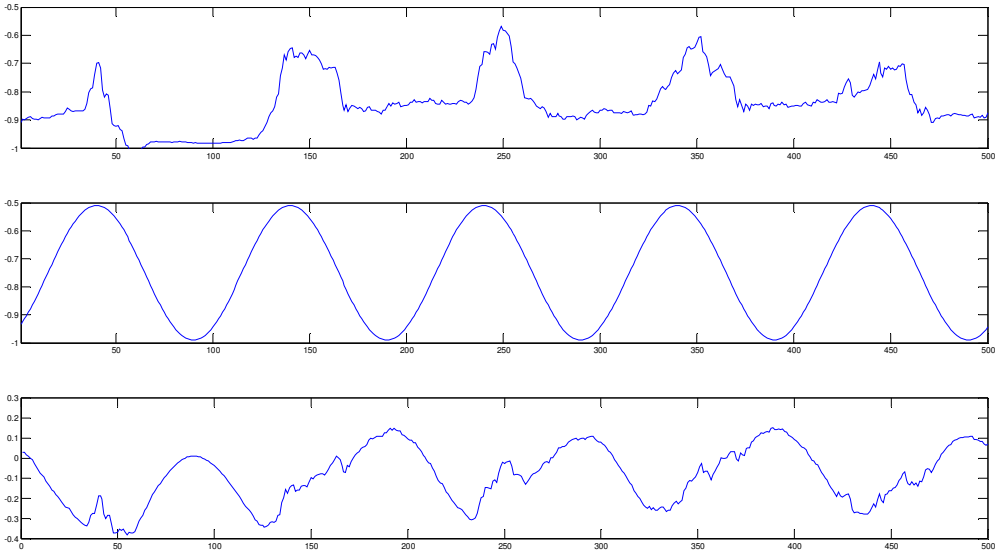
To estimate these parameters using Bayesian techniques, the output would be fitted against the known input signal, zero mean Gaussian white noise. As the output conformed or disagreed with the assumed model, the distribution of the presumed alphas would change until the probability distribution would center on the correct value. Updating the probability of model given the data would present a cleaner result than the noise seen in the above ML estimates.

4. Stationarity.

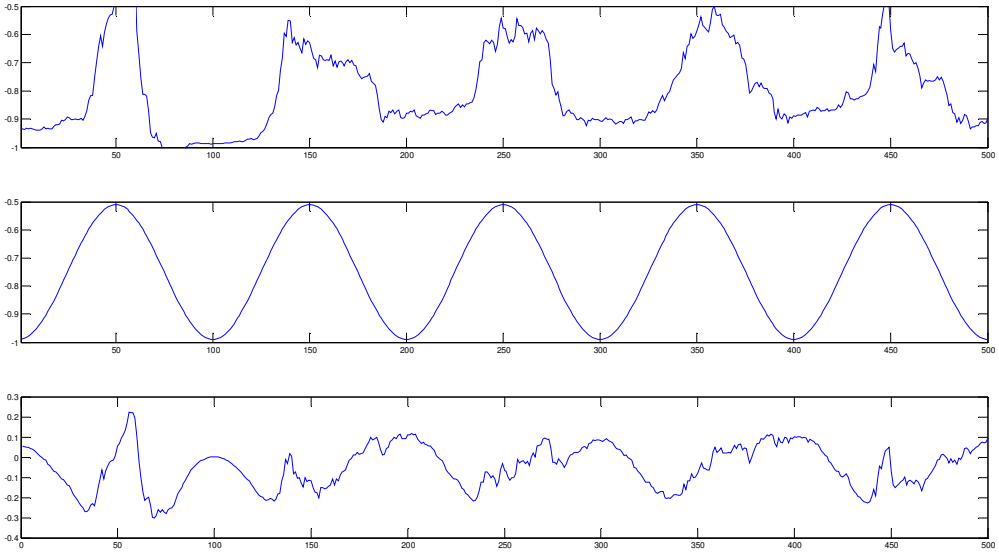
Allowing the coefficient to vary with time, in an equation such as $H(z) = 1-az^{-1}$ where $a = 0.75 + 0.24\sin(2\pi n/100)$, produces an output as follows. The top plot is the zero-mean Gaussian input and the bottom plot is the output signal.



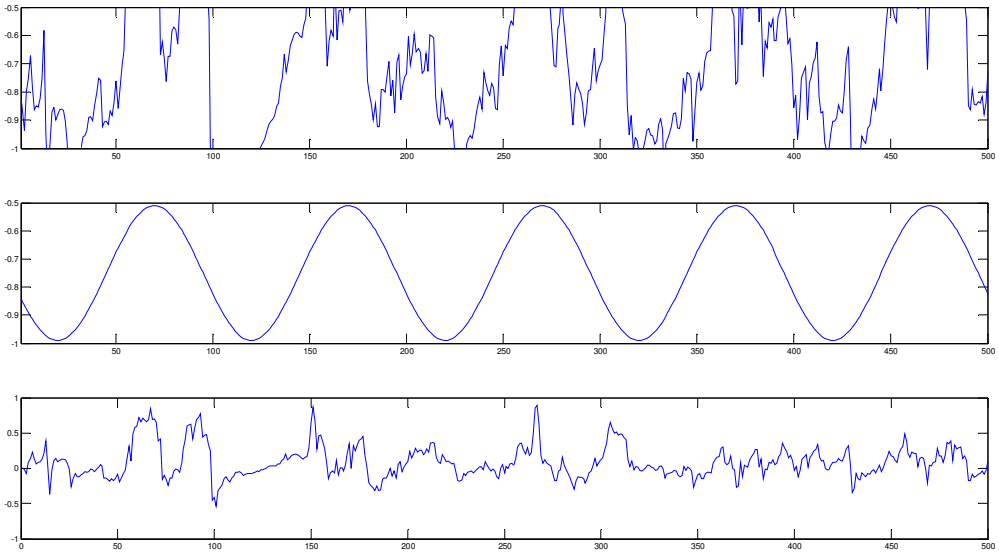
An analysis of the autocorrelation functions output as various window sizes are used to compute the coefficient of the filter. As the window size is varied, and the function works itself through the output signal it is clear that increasing the window size is not the panacea. Below is a window of size 70 points.



The top plot is the result of the autocorrelation function, while the middle plot is the injected coefficient. The plot on the bottom is the difference between the two signals. The results of the autocorrelations R(1)/R(0) does not match up well with the input signal at all. Compare this result to when the number of samples in the window is set for 50.



Here with a window of 50 samples, the autocorrelation calculation returns something that does resemble a periodic signal. The difference between the two signals gives hope for this entire process to work out. Far worse is a filter with a window of 10 samples that produces gibberish for the estimation of the coefficient over time.

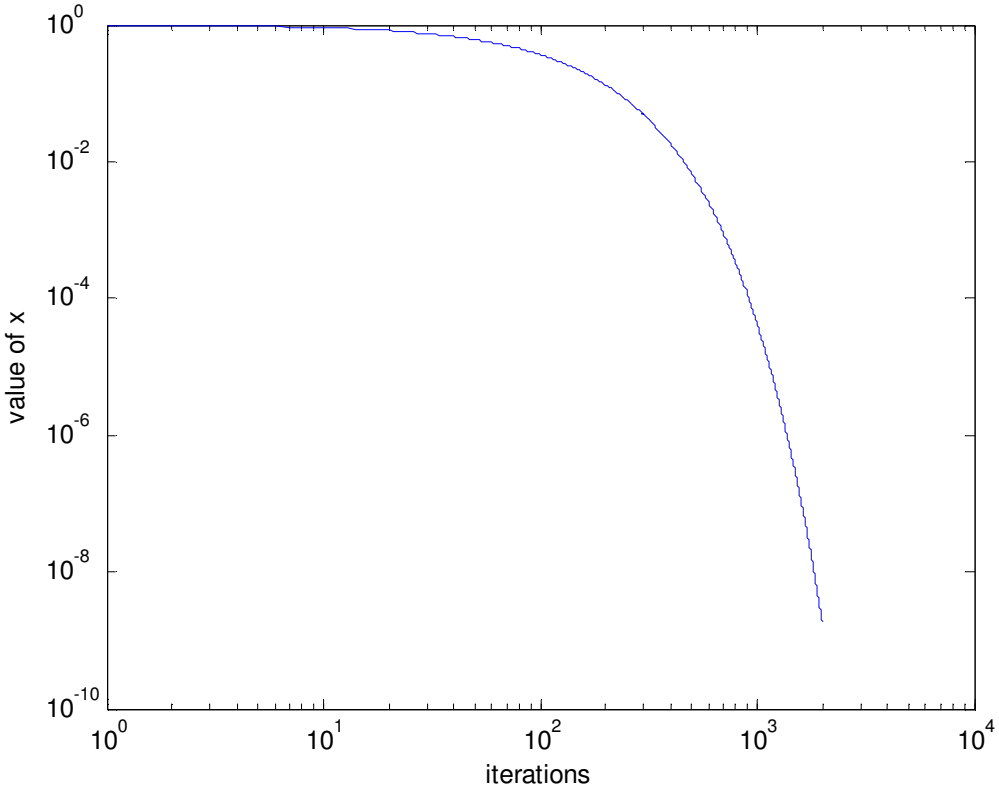


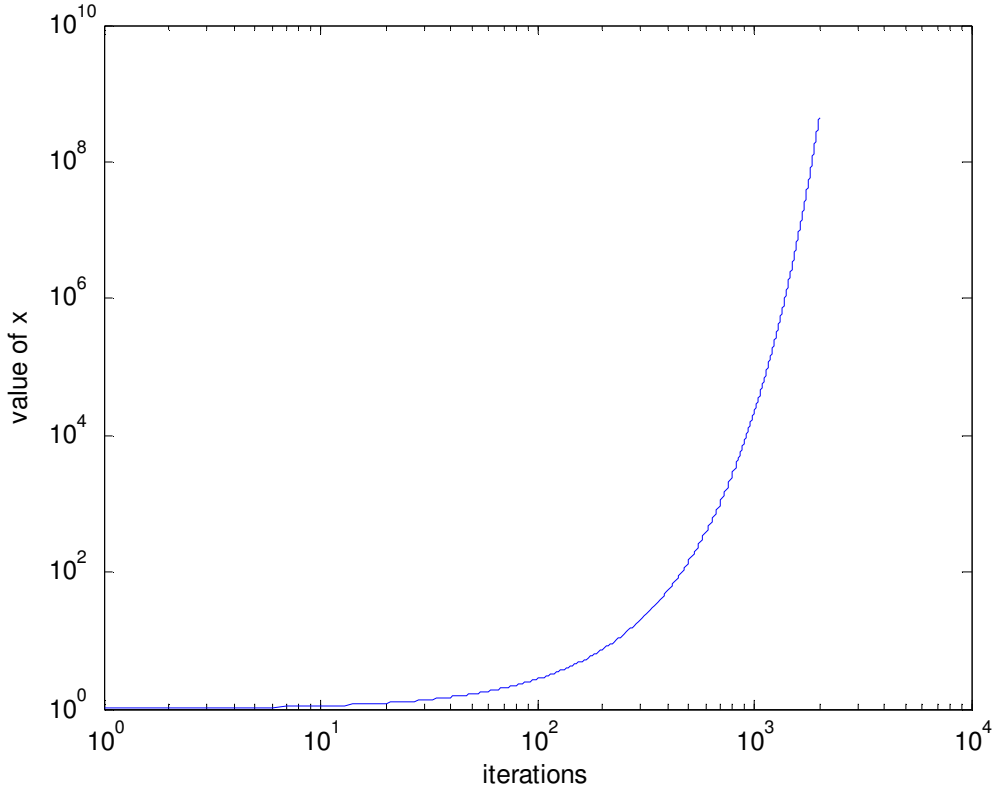
This result does not even stay within the window for what the coefficient should be and produces a lot of errors. Window sizes below 10 produce very wild results, but my initial assumption was that the window should not exceed the rate it is sampled at, which in this problem is 100Hz. It is clear that the result should track the coefficient when the proper size of

the filter, n, is found. However, the best value of n discovered appears to be that of a 50 sample window, which does not fully track the input.

5. Stability.

   Given $x[n] = a^n u[n]$ where a=0.99 and 1.01 and is sampled at outputs of n=0, n=100, n=1000 for a given window, produce 2x2 covariance matrices. The chosen window is 50 samples with the two plots of the initial functions represented below in double log plots.





The function with the coefficient that is less than 1 is driven down to zero, while the function with the coefficient greater than 1 balloons up to what would be infinity if given enough data points. In these simulations only 2000 data points were used to build the outputs.

covar_0_099 =
1.0000 0.9849
0.9849 0.9701
covar_100_099 =
1.0000 1.0101
1.0101 1.0203
covar_1000_099 =
1.0000 1.0101
1.0101 1.0203

The above results are from the covariance matrix taken over 50 points from 0, 100, and 1000 in the data set. The values have been normalized, C/C(1,1), to highlight the relationship between the terms. They are part of the a=0.99 coefficient growth that reaches zero at large values of n. Not to be outdone, the growing term where a=1.01, results in values where each entry is less than that of the C(1,1) aside from when the function first starts out.

covar_0_101 =
1.0e+08 *

```
    0.0000    0.0003
    0.0003    8.9707
covar_100_101 =
    1.0000    0.9901
    0.9901    0.9803
covar_1000_101 =
    1.0000    0.9901
    0.9901    0.9803
```

The results of both matrices show that the division of the terms in the first column produces a factor that scales the row 1 terms from the row 2 terms. However, for the growing term the non C(1,1) terms decrease and become less than the leading term. The opposite is true for the decay output as the non C(1,1) terms grow larger than the leading term.

As the term C(1,1) is actually the R(0) autocorrelation term, this shows that one of these functions keeps gaining in energy as the others lose energy for the exponentially growing function. However, the decaying function is dominated by the growth of the other correlations and NOT the C(1,1) term. This indicates that if the leading term grows faster in relation to the other terms, the function is unstable and does not result in a finite integral.