Environmental Data Analysis with *MatLab, 2nd Edition*

Copyright 2015 by William Menke and Joshua Menke

All Rights reserved

Solutions to Problems

Chapter 1

1.1 See *MatLab* script prob01\_01.m.

1.2 Note that the command, C = [a, b], creates an *N×2* matrix, **C**, from two length-*N* column-vectors. See *MatLab* script prob01\_02.

1.3 Note that *MatLab* has many alternative ways to create a small matrix. Here we initialize the whole matrix to zero, and then individually set each non-zero component. This is not the cleverest method, but it works. See *MatLab* script prob01\_03.

1.4. See *MatLab* script prob01\_04.

1.5 This problem can be solved in many ways. Here we use the find() command to build a column-vector, dbad, that contains the indices of all the points with negative discharge, and then output the length of that vector. See *MatLab* script prob01\_04.

Chapter 2

2.1 By examining the graph, it is clear that the troughs in the temperature oscillations occur very nearly at the start of each year (that is, in January), indicating that the cycle is annual. See *MatLab* script prob02\_01.

2.2 The largest hourly change in temperature is computed from the finite difference approximation of the derivative, d*d*/d*t*. The tricky part of the script is excluding the bad data, since pairs good data are needed to compute the derivative. Four types of bad data are detected: missing data (pairs of data separated in time by more than *t*), cold spikes (*d<40* °C), hot spikes (*d>38* °C) and drop outs (*d=0*). These conditions are tested within the for loop, with bad data being skipped using the continue command. An alternative formulation, which omits the loop, uses the find() command. See *MatLab* scripts prob02\_02 and prob02\_02b.

2.3 A modified version of script eda02\_03 is used to plot representative summer and winter segments of the time series (see plots below). In both cases, the cycle is asymmetric, with a sharp rise in temperature in the morning and slower cooling in the evening. The amplitude of the diurnal signal is larger in the summer than in the winter (*10-12* °C in summer compared to *6-8* °C in the winter). The morning rise in temperature occurs at a later time of day in the winter than in the summer, reflecting the shorter length of day in the winter. See *MatLab* script prob02\_03.

Summer:



Winter



2.4 Changes needed to adapt the eda02\_03 code to the Neuse River Hydrograph include the filename, the title and labels of the plots and the length of data plotted in Figure 2. See *MatLab* script prob02\_04.

2.5 See *MatLab* script *prob02\_05*.

Chapter 3

3.1 We determine the constant, *c*, in the probability density distribution, by requiring that the distribution have unit area, :

:

The mean, , and variance, *m2*, are:

We check these values against numerical calculations. See *MatLab* script prob03\_01.

3.2. The probability density distributions for the *=5* and *=10* cases are shown at the right. Note that the *=10* case is narrower. The respective means are approximately *0.19* and *0.10* and the respective variances are approximately *0.03* and *0.01*. Since *p(d)* has only one parameter, **, its mean and variance cannot be controlled separately. See *MatLab* script prob03\_02.

3.3 In this problem, we have one variable, *d*, related to another variable, say, *E=|d|½*. We use the formula, *p(E) = p[d(E)] |*d*d/*d*E|.* In this case, *d=E2* and d*d*/d*E=2 E*. The probability density distribution, *p(d)*, is given as having zero mean and unit variance. Since only the absolute value of the *d* matters, we need only to consider *d*’s that are greater than zero, so the Normal distribution for *d* has twice the usual amplitude:

A grey-shaded version of the probability density distribution, *p(E)*, is shown at the right. The mode is approximately *0.84*; the median is, *0.82*; the mean, *0.82* and the variance, *0.12*. See *MatLab* script prob03\_03.

3.4. We use Bayesian inference (Equation 3.16) to solve this problem. A fraction, *0.014*, of the general population die of pancreatic cancer and a fraction, *0.986*, die of something else. Before the test is made, the best estimate of the probability that the person died of cancer is *0.014*. The fraction of positive tests is the fraction of people who died of cancer and who correctly tested positive plus the fraction of people who died of something else but who incorrectly tested positive, that is, *0.014×0.99+0.986×0.01 = 0.0237*. Of these, only *0.014×0.99=0.013*8 died of cancer. So the probability that the person died of cancer, given the positive test result, is *0.0138/0.023≈0.58*, or *58%*. While our estimate of the probability that the person died of cancer increases by an order of magnitude, from *1.4%* to *58%*, we are still not particularly sure that cancer was the cause of death (that is, the probability is much less than *95%*). This scenario illustrates the problem of trying to diagnose a rare condition with an imperfect test. The number of true-positives is overwhelmed by the number of false-positives. [Thanks to Brian Mapes and his class for pointing out an error in the original solution!]

3.5 The sum, *m1=d1+d2*, and difference*, m2=d1d2*, of the data is: of the form **Md**=**m**:

Assuming uncorrelated data of equal variance *d2*, the covariance of the model parameters is **C**m = 2 *d2* **I**. The sum and difference are uncorrelated and both have the same variance, 2*d2*.

3.6 If the data are uncorrelated and with equal variance, then the covariance matrix for the data is of the form, **C**d = *d 2***I**. The covariance of the **m** is then is **C**m = *d2* The *m*’s are uncorrelated only when **C**m is diagonal, and this happens only when all the rows of the matrix, **M**, are mutually orthogonal.

Chapter 4



4.1 The matrix, **G**, has the form:

The covariance of the model parameters is given by **C***m*=*d 2* [**G***T***G**]1. The error in the *j*-th model parameter, *mj*, is the square root of *j*-th diagonal element, [**C***m*]*jj* (see figure at right). The error grows with the index, *j* (though more slowly than linearly), since the value of the *j*-th model parameter is dependent upon all of the first *j* measurements. See *MatLab* script prob04\_01.

4.2. This equation can be linearized by taking the logarithm:

where *d’i*=ln(*di*) and *m’1*=ln(*m1*). However, note that least squares methodology is only appropriate for solving problems were the error is Normally distributed. Suppose that the error in d’ is Normally distributed:

The transformation of variables is *d*’=ln(*d*), so d*d’*/dd = 1/d and the transformed probability density distribution is:

Least squares is only appropriate when the original data, *d*, obeys this *log-normal* distribution.

4.3. A) The matrix, [**G***T***G**]*ij* = **c***(i)T***c***(j)*, where **c***(i)* is the *i*-th column of **G**. B) The matrix, **G***T***G**, is diagonal when all the columns of **G** are mutually perpendicular, that is, when their dot products are all zero. C) If the data are uncorrelated with equal variance, *d2*, the covariance matrix for the model parameters is **C***m*=*d2*[**G***T***G]1**, which in this case is diagonal. D) Thus, the least squares solution **m**est = [**G***T***G]1G***T***d** requires only a trivial matrix inverse (since the inverse of a diagonal matrix is a diagonal matrix with elements that are the reciprocals of the original). It is much easier to compute than the general case. E) In the straight line case:

The off-diagonal elements of **G***T***G** are only zero when the mean of **x** is zero, which can only happen when the data are centered on the origin. The model:

is a better formulation of the straight line problem, since the model parameters, *m’1* and *m’2* are guaranteed to be uncorrelated.

4.4. In all three cases, only the *2*nd coefficient is significantly different from zero. See *MatLab* script prob04\_04.

4.5 The modifications lead to a reduction of the total error, E, from 3.3099e+006 to 2.8542e+006, mainly though fitting the diurnal cycle. However, the estimate of the slope and its variance is not significantly changed. See *MatLab* script prob04\_05.

Chapter 5

5.1 The prior information, that “negative densities are nonsensical” cannot be implemented through a linear constraint of the form **Hm**=**h**. (Advanced techniques are available for implementing *positivity constraints*, but they are not discussed in this book).



5.2 Inconsistent prior information is not a problem. The method is minimizing the error associated with the prior information, not fitting the prior information exactly. Thus, it merely ‘splits the difference’ (see figure at right). See *MatLab* script prob05\_02.

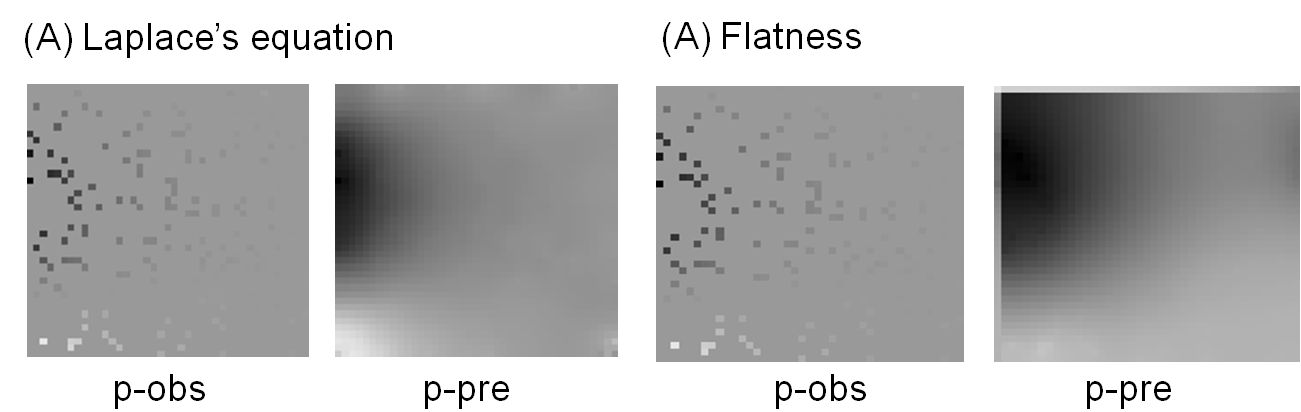
5.3 The method (see figure below) smoothes out the edges of the data gaps, but (not surprisingly) cannot fill in big sections of missing data in a convincing way. See *MatLab* script prob05\_03.

5.4 The *MatLab* script prob04\_04b is a version of eda05\_08, modified to write out nine different pressure files, each computed with a different combination of  and d. The MatLab script reads and processes these files in sequence, producing the results shown in the figure, below. Note that as  is increased, the pressure, *p(x,y)*, becomes concentrated at small values of *y* (the left portion of the grid). Since the data are sparse, the data contain little useful information about the behavior of short wavelength features. Thus method tends to fit the noise at the higher noise levels.

5.5 An important difference between the pressure problem and the velocity problem is that pressure is a scalar while velocity is a vector. While the pressure problem has one model parameter, *p(x,y)*, per grid point, the velocity problem has two, *vx(x,y)* and *vy(x,y).* The number of model parameters is *2IJ*, where the grid is of dimensions, *I×J*. In the script, these model parameters need to be organized in two alternate ways, in two *I×J* matrices, Vx and Vy, and as a single length *2IJ* column-vector, mest, of model parameters (with Vx unwrapped into the top half of mest and Vy into the bottom half). The first *2IJ* rows of **Fm**=**h** are the data equations, *mi = vxobs(xi,yi)* and *mi = vyobs(xi,yi)*. The divergence equation is satisfied on the interior *(I2)×(J2)* grid points. It corresponds to the finite difference equation:

One the edges of the grid, the following *2I+2J* equations are satisfied

The data equations have one non-zero element of **F** per row, the divergence equations, *4*, and the boundary equations, *2*. The data equations need to be multiplied by the certainty of the data, *d1* and the remaining equations by the certainty of the prior information, *h1*. The total number of equations (rows in **F**) are *2IJ+(I2)(J2)+2I+2J*.

5.6. The combination of flatness prior information with and smallness prior inforation with produce a prediced pressure that is smooth and matches the observations fairly well. However, the error is not nearly as small as for the Laplace’s equation prior information (529.44 

compared to 0.07), presumably because the pressure field from which the data are drawn really do satisfy Laplace’s equation.



Chapter 6

6.1 The Neuse River discharge, *d(t)*, is shown at right, as it its derivative, d*d*/d*t*, calculated by both finite differences (fd) and the Fourier transform (fft). See *MatLab* script prob06\_01.

6.2 The Fourier transform of *d(t)=sin(0t)* is:

As can be seen by inserting this formula into the inverse transform:

Note that both the transforms of both *cos(0t)* and *sin(0t)* consist of a pair of spikes at ±0. Only the amplitude of the spikes differs.

6.3 The power spectral density, *s2(f)*, of the random time series contains power at all frequencies (see figure at right). The power spectral density of the time series that has been smoothed by the moving window average lacks high frequencies. The longer the average, the more high frequencies are suppressed. The low frequencies seem to be little affected by the averaging. See MatLab script prob06\_03.

6.4 The Fourier transform of the Normal curve is computed by the two methods described in the problem. In Method 1, times less than zero are wrapped around to the right hand side of the time interval. In Method 2, the Normal curve is centered at time, *t0=T/2*, where *T* is length of the time series, and its Fourier transform is multiplied by the phase ramp, *exp(it0)*. Results are shown in the figure at right. In both cases, the imaginary part of the transform is zero. See MatLab script prob06\_04

6.5 The Black Rock Forest dataset was cleaned by filling in missing data points, which were given the value of zero, and zeroing cold and hot spikes. The code for filling in the missing data needs some explanation:

D = floor(((traw(2:Nraw)-traw(1:Nraw-1))/Dt)+0.5);

m = [1, 1+cumsum(D)']';

N = max(m);

tc=tmin+Dt\*[0:N-1]';

dc=zeros(N,1);

dc(m)=draw;

The raw time and data values, as read in from the file brf\_temp.txt, are in the column-vectors, traw and draw, respectively. The time difference between neighboring samples in draw is usually Dt=1/24 days. However, since some data are missing, the time difference is sometimes a multiple of Dt. The column-vector, D, is the number of samples between data, computed by rounding of the time difference to the nearest multiple of Dt. The cumulative sum of these increments, m, gives the index of the data into a time series, dc, with uniform time increment, Dt. A corresponding time column-vector, tc, is also computed. All missing data values are set to zero, as are (elsewhere in the script) all hot and cold spikes.

These zeros will affect the power spectral density estimate to some degree, but we proceed with the calculation, anyway, hoping for the best. The power spectral density (see figure below) has spectral peaks at frequencies of 1/365, 1, 2, 3, 4, and 5 cycles per day. The peaks at frequencies higher than one day are due to the diurnal cycle having a 1-day periodicity while not having a sinusoidal shape. They modify the shape of the diurnal sinusoid.

.

Chapter 7

7.1 The first element of the convolution is:

The second element is:

The fourth element is:

The fifth element is:

The seventh element is:

So the convolution is the triangular time series, *[1, 2, 3, 4, 3, 2, 1]T*.

7.2. Beyond filter lengths of about *8*, the error decreases only very slowly with filter length and is very flat beyond a length of *70* samples (see figure, below). The error does decrease about *5%* between filter lengths of *40* and *50* samples, a time interval where the longer filters have a small feature. This behavior indicates that the feature does noticeably improve the prediction. See *MatLab* script prob07\_02.



7.3. The filter, **f***=[0, 1]T*, with z-transform, *f(z)=z* , delays a time series by one sample. Note *f(z) (1 + az + bz2 + cz3+…) = (0+ z + az2 + bz3 + cz4+…) → [0, a, b, c, …]T*.

7.4 The amplitude of the first coefficient of **g** is reduced by steps of *13%*, with the first step corresponding to no reduction. The corresponding inverse filter, **g***inv*, is computed and displayed in the figure below, with the first filter coefficient, *g1*, and number of roots not outside the unit circle displayed above each graph. The filter length increases as the amplitude of *g1* decreases (see the graph, below). When the filter becomes non-minimum phase, the coefficients of *giinv* grow with index, *i*. See *MatLab* script prob07\_04.



7.5 We start with:

The first question is how to normalize this filter to unit area. For a continuous filter, the proper normalization is ** since the area under *(1/)exp(t/)* is unity. The Riemann sum approximation implies a normalization of approximately *t/* for the discrete case. An exact normalization can be computed by noting that the summation, *1+c+c2+c3+…* , is a geometric series and is equal to *1/(1c)*. Thus, the exact normalization is *(1c)*. Note that 1exp(t/) ≈ t/ when t. Then by analogy to Equation 7.23, *u=[1c, 0]T* and *v=[1,c] T*. These formulas are tested in *MatLab* script prob07\_05.

7.6. Representative portions of the original (top) and filtered Neuse River Hydrographs are shown below. The larger the , the smoother the time series, and consequently, the lower the peak amplitude. See *MatLab* script prob07\_06.



Chapter 8

8.1 Since **S** =**UV***T*, **SS**T = (**UV***T)(* **UV***T) T =* **UV***T***VU***T* **= U******U***T*, since **V***T***V=I**. Postmultiplying by **U** yields (**SS**T)**U** = **U***.*, since **U***T***U=I**. This is the matrix form of the algebraic eigenvalue problem for the matrix, **SS**T (see Equation 8.11). Thus **U** are eigenvectors of **SS**T.

8.2 Least squares can be used to find the coefficients, *m*i(k), for expressing the *k*-th specified factor, **f***(k)s*, in terms of the *P* original factors, **f***(j)*. The equation is:

The coefficients,**m**(k), for each specified factor can be solved for separately. The result will not necessarily equal the specified factor but it will approximate it in the least squares sense.

8.3 The power spectral density of the loadings of the first *12* factors is shown in the accompanying figure. The first *4* have negligible power at a period of one year. Several of the others (particularly *8* and *10*) have well-defined annual peaks. Se *MatLab* script eda08\_03.

8.4 The first six factors (or so) have significant amplitude. The first factor, the typical diurnal cycle, has one oscillation per day. Its maximum occurs at hour *14*, that is, somewhat later than mid-day. The second factor also has one oscillation per day, with a maximum at about *6* hours. Presumably, when added to the first, it shifts the time of the thermal maximum. The other factors have increasingly more complicated patterns. The loadings (coefficients) of the first few factors have an obvious annual cycle. The loading of factor 1 is largest in summer, confirming a conclusion made earlier (in Problem 2.3) that the diurnal cycle is largest in summer. The first two factors and the first two loadings are shown in the plot below. See *MatLab* script prob08\_04.





8.5 Lead, an important pollutant, has four stable isotopes, 204Pb, 206Pb, 207Pb, 208Pb, which occur in different proportions in lead derived from different ores. Suppose that dangerous levels of lead contamination are discovered in a region where two sources of the pollution have been hypothesized (e.g. natural lead weathering out of local rocks and lead from a lead-acid battery manufacturing plant). A set of soil samples can be collected from a two-dimensional grid of sites in the region and the isotopic composition of their lead determined by mass spectrometry. Factor analysis can then be used to determine the number, *P*, of factors, which in this case have the interpretation of lead sources. Up to *P=3* sources can be identified, since three isotopic ratios can be formed from the four lead isotopes. In this case, we might find that *P=2*, since two sources of the lead are suspected. If this is the case, then the first SVD factor would represent the average isotopic composition of the lead and the second the deviations from the average. Lead from samples located near a lead source would probably mostly be derived from that source. If so, the loadings of the second factor could be used to infer the location of the lead sources, since, when plotted on the grid, they would vary spatially in a systematic way, being large and positive near one source and large and negative near the other.

Chapter 9

9.1 The results are that temperature lags solar radiation by 2 hours. See the figure below and MatLab script prob09\_01.



9.2 We start with **f**=**Sh** (the matrix form of *f(t)=s(t)\*h(t)*), along with the error propagation rule **C***f*=**SC**h**S***T*, where the **C**h is the covariance matrix of **h**. Then, assuming a time series with zero mean, **C**h = **A**h ≈ **HTH**. Here **H** is the convolution matrix formed from **h** and **A**h is the autocorrelation matrix of **h**. Then, **A**h= **SHTHS***T*. Now read each matrix as a convolution, noting that the transpose of a convolution matrix reverses the sense of time. The autocorrelation, ah(h), of h(t) then satisfied:

ah(t) = s(t) \* h(t) \* h(t) \* s(t) = [s(t)\*s(t)] \* [h(t)\*h(t)] = [s(t)⋆s(t)] \* [h(t)⋆h(t)] = as(t) \* ah(t)

where as(t) is the autocorrelation of *s(t)*. This is the same rule as is given in Equation 9.24 for the autocorrelation of a filtered time series, so the two methods agree.

9.3 The autocorrelation of the Reynolds Channel chlorophyll data is shown in the accompanying figure. The correlation falls off quickly for lags of a week, and then more slowly for lags of a few weeks, crossing zero at a lag of 78 days. At larger lags, the autocorrelation oscillates around zero with a period of about two years. If an annual oscillation, it is small.



9.4 The power spectral density of the raw and Hamming windowed Neuse Hydrograph time series are shown in the accompanying figure. The major peaks are visible in both versions, but some of the minor peaks are less clear in the windowed one. The frequencies at which the peaks occur are unchanged. See *MatLab* script prob09\_04.



9.5 As we hypothesized in previous analyses of the Black Rock Forest temperature dataset, the amplitude of the diurnal cycle varies systematically during the year, being largest in summer. See *MatLab* script prob09\_05.



Chapter 10

10.1 Suppose that we linearly interpolate *N* data, **d**, evenly-spaced between two observed data, **d***obs=[dA*, *dB]T*. The equation for linear interpolation (Equation 9.1) can be written in matrix form:

By the usual rules of error propagation, **C**d = *d*2**MM**T, where we have assume that the observed data are uncorrelated with variance, *d*2. Then,

This matrix is shown for the case, N=50, *d*2=1 in the accompanying figure. Note that the variance is largest for *d1* and *dN*, were it equals unity (that is, the same variance as the uninterpolated data), and declines to *d*2=0.5 at the center (where it the interpolated data is the average of the two observed data). All the interpolated data are positively correlated with one another, except the first and the last, which are uncorrelated. See *MatLab* script prob10\_01.

Intuitively, we might expect the covariance to be largest between the samples, but in this analysis it is smallest. This is because we have not taken into account the *uncertainty* of the prior information that the data varies linearly between the data points.

10.2 A plot of the cubic spline interpolation of the Reynolds Channel Water Quality data is shown in the figure below. It contains large swings in the stretches where the observed data are absent (e.g. air temperature, chloroplyll) and in this sense is worse than the linear interpolation.



10.3 The Kriging result is shown in the figure, below. As expected, the interpolated function is very rough for small values of *L*. The results do not seem especially sensitive to the value of L for when *L* is in the range of *4* to *30*, but by *L=100* the results are poor.



10.4 The graphs below of the two-dimensional cosine function, *F(x,y)*, and its power spectral density, , can be compared to the untapered versions shown in Figure 9.19. The spectral peaks of the tapered version are wider, but the side lobes are significantly suppressed.



10.5 The two-dimensional power spectral density of the 9th image of the CAC Sea Surface Temperature dataset is shown below. The sea surface temperature data have only long wavelength features, so the high spectral densities are concentrated near the origin (at the center of the image). See *MatLab* script prob10\_05.



Chapter 11

11.1. We expand around by noting that and so . Then:

We plot the function (grey) and its approximation (black), below. The error is less than in the interval .



11.2. We expand the functions and around means and using number approximations:

so

Thus:

We check this calculation by comparison to a numerical simulation in the accompanying script.

11.3. The accompanying script fits both diurnal and annual periods. A comparison of observed (black) and predicted (grey) temperatures for a summer month shows that the predicted daily oscillation has about the same amplitude and phase as the observed. However, the posterior is reduced only from 5.6 degrees C (when just a annual frequency is included) to 5.2 degrees C.



11.4. Suppose that the function is evaluated times. Then the total time needed to evaluate it is . If the table has entries, then the time needed to compute it is and the time needed to perform lookups is . The time is reduced by when:

Solving for yields:

11.5. The accompanying script initializes a network with two towers, one at the maximum of the function at the other at its minimum. The training reduces the error from to 0.12.



11.6. Once trained, two tower function neural net does an excellent job representing the parabola, reducing the error from to .



11.7 The script eda\_setup\_11\_07.m creates a file datafor1107.txt containing a -point data set in which the coefficient linearly increases from at the start to at the end. The script eda\_problem\_11\_07 trains a neural net on the first data points. It then predicts the data in two ways: using the neural net, as is (grey curve) and by re-training the neural net using a single iteration at every time step (black curve). The error of the un-retrained network grows with time, but the error of the re-trained network remains approximately constant. This result demonstrates that a single iteration is sufficient to keep the neural net up-to-date with the changing value of the coefficient . 

Chapter 12

12.1 A) The Null Hypothesis is that any difference between the two means arise from random variation alone. B) The MatLab script prob11\_01 was used to calculate the following information:

year 1: N 10 mean 32.392000 sigmad 0.601845 sigmam 0.190320

year 12: N 10 mean 33.756000 sigmad 0.181488 sigmam 0.057391

t-est -6.861678 degrees of freedom 10.623382

P(|t|> 6.861678 = 0.000032

The absolute value of the *t*-statistic for this problem is, *|test|=6.86*. The probability that *t* will equal or exceed this value due to random variation is only *0.0032%*. C) The Null Hypothesis can be rejected to greater than *99.9%* confidence.

12.2 The Null Hypothesis is that differences in the total error of the prediction error filters of different lengths arise from random variation. The MatLab script prob07\_2 was modified to compute the *F* statistic and to assess its probability for every pair of filter lengths. The accompanying figure shows the significance level of the pairs (left image) and the pairs for which one filter of the pair has significantly different error than the other (right image). Filters of length *>2* are significantly better than filters of length *2*, to greater than *95%* confidence. No filter with length *>2* is significantly better or worse than any other filter length>2 filter. This finding may be deceptive, since the error time series, *e(t)*, does not really resemble uncorrelated, Normally distributed noise with zero mean and uniform variance. See *MatLab* Script prob11\_02.

12.3 The probability density distribution. *p(d)*, for datum, *d*, is assumed to be Normal:

Now consider the transformation, , and note d*d/*d*Z* = . The transformed probability density distribution is:

By inspection, this distribution is Normal with zero mean and unit variance.

12.4 A) Figure 10.6B has about *15* peaks, each about *1* sample wide, that exceed the 95% confidence threshold. The number of samples is *1024*, so the number of positive frequencies is *512.* Thus, about 3%samples exceed the threshold. This is less than the expected number, 5%, presumably because of random variation. B) The largest peak has an amplitude of about *s2=8.1*. The quantity *s2/c*, with *c=0.634* is chi-squared distributed with *2* degrees of freedom. We use MatLab to evaluate the expression, chi2cdf(8.1/0.634,2), which equals *0.9983*. Thus the highest peak is significant to the 99.83% level.

12.5 The Null Hypothesis is that the spectral peaks arise from random variation of an uncorrelated, Normally distributed random time series with constant variance. The *95%* and *99.999%* confidence intervals are superimposed on the power spectral density (PSD) of the Neuse River hydrograph (see graph below). Most of the part of the spectrum in the frequency range *0.00-0.04* cycles per day is significant to the *95%* confidence level (frequencies near the Nyquist frequency of 0.5 cycles per day are not plotted). Several of the peaks, and in particular the annual peak (0.0027 cycles per day) are significant to better than the *99.999%.* confidence level. Mostly, this test is verifying what is fairly obvious: that the hydrograph has much more power at low frequencies than would be expected of an uncorrelated, Normally-distributed time series. See *MatLab* script prob11\_05.