SIMULTANEOUS CONFIDENCE BANDS FOR NONPARAMETRIC, POLYNOMIAL-TRIGONOMETRIC REGRESSION ESTIMATORS

Jonathan W. Duggins

A Thesis Submitted to the
University of North Carolina at Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Science

Department of Mathematics and Statistics

University of North Carolina at Wilmington

2003

This thesis has been prepared in the style and format

consistent with the journal

Journal of the American Statistical Association.

TABLE OF CONTENTS

ABSTRACT

The goal of this paper is to present a method for construction of simultaneous Scheffé confidence bands for nonparametric prediction functions. The family of nonparametric functions studied here are of the polynomial-trigonometric series type, with estimation of the model parameters undertaken in the standard least-squares framework. After the method is set forth the performance of the procedure is studied via Monte-Carlo simulations.

1 INTRODUCTION

Typically when confronted with the idea of regression one immediately thinks of fitting some specific model to given data. Traditionally, this is done by taking the data to be (t_i, y_i) for $1 \leq i \leq n$ and assume data follow a model of the form $y_i = f(t_i) + \epsilon_i$. Here ϵ is a vector of *iid* error terms with common variance σ^2 and $f(\cdot)$ is unknown.

One must now decide whether to proceed via parametric or non-parametric analysis methods. Parametric regression depends on assuming a particular form of $f(\cdot)$, say linear or quadratic. On the other hand, non-parametric regression simply assumes that the predictor function is an element of some infinite dimensional function space. While parametric is the most popular choice it is, in general, only valid if the data actually follows the designated model. Thus, if data collection is done in order to ascertain the nature of the relationship, it would seem inherently restrictive to choose parametric regression since the data in question may not follow the model chosen. Thus, often it is reasonable to look further at a non-parametric model. As a result of this the model is still $y_i = f(t_i) + \epsilon_i$ but now $f(\cdot) \in \mathbb{F}$, some as yet unknown infinite dimensional function space.

2 DETERMINING $f(\cdot)$

The nature of $f(\cdot)$ must be determined before proceeding to make inferences about the presence of a relationship in the data. One approach is to try to write $f(\cdot)$ as a linear combination of known elements in the same function space. One may then look for a basis for $\mathbb{F}$, say $\{x_k\}_{k\geq 1}$, and write $f = \sum_{k=1}^{\infty} \beta_k x_k$. Note that $\{x_k\}$ is necessarily infinite since $\mathbb{F}$ is an infinite dimensional function space. Unfortunately, this sequence may not be available for every class of functions so a particular function space must now be chosen. For concreteness take the space to be $L_2[a,b]$, the set of all square integrable functions on the interval $[a,b]$. To describe the nature of the basis elements in this space a few definitions are needed.

Definition 1 *The norm of a function $g(t) \in L_2[a,b]$ is defined as*

$$||g(t)|| = \left\{ \int_a^b [g(t)]^2 dt \right\}^{1/2}.$$

Definition 2 *The inner product of two functions $x_1, x_2 \in L_2[a,b]$ is defined to be*

$$< x_1, x_2 >= \int_a^b x_1(t) x_2(t) dt.$$

Definition 3 *Two functions $x_1, x_2 \in L_2[a,b]$ are said to be orthogonal if $< x_1, x_2 >= 0$.*

Definition 4 *A sequence of functions $\{x_k\}_{k\geq 1}$ is said to be orthonormal if the x_k are pairwise orthogonal and $||x_k|| = 1$ for all k.*

Definition 5 *A sequence of functions $\{x_k\}_{k\geq 1}$ is said to be a complete orthonormal sequence (CONS) if $< f, x_k >= 0$ for all k implies $f \equiv 0$.*

Then if a CONS can be found it is necessarily a basis for $L_2[a,b]$ (Eubank 1999).

As before, the function is $f(t) = \sum_{k=1}^{\infty} \beta_k x_k(t)$, but now in order to proceed further one must decide on a particular sequence of functions. There are two basic ways to build the sequence, either by choosing a polynomial or trigonometric basis. The former may be completed with the use of the Legendre Polynomials obtained via Gram-Schmidt orthonormalization. Unfortunately the matrix associated with the polynomial basis is usually ill-conditioned which produces numerical difficulties. One may choose then to try using a trigonometric basis which can be formed in one of three ways. The first is to use both sine and cosine functions. For example, take $x_1 = 1,\ x_{2k}(t) = \sqrt{2}\cos(2k\pi t)$, and $x_{2k+1} = \sqrt{2}\sin(2k\pi t)$ for $j \geq 1$ on the interval [0,1]. Alternatively, the functions may be defined in terms of either sine or cosine individually. The most common trigonometric sequence is formed using solely the cosine function since its performance at the boundaries of the interval $[a, b]$ is superior to the other two. However, it still does a rather poor job there as the boundary bias may be substantial. The solution is to combine the two methods in some fashion. The method used here was chosen such that a fixed number of polynomial functions are employed to improve the boundary behavior and allow the remainder of the functions to be cosines in order to achieve an acceptable compromise between numerical complications and boundary bias (Eubank 1999). In particular, choose $x_1 = 1,\ x_2 = t - \frac{t^2}{40},\ x_3 = \frac{t^2}{2}$ and $x_k = \sqrt{2/(b-a)}\cos[(k-3)(t-a)\pi/(b-a)]$. From this it can be seen that $f(t)$ is a linear combination of known functions all which lie in $L_2[a, b]$ as was originally proposed.

Now that this has been established, estimates of β_k for $k \geq 1$ must be found. But it is known that $\beta_k \to 0$ as $k \to \infty$ so the infinite series, $f(t) = \sum_{k=1}^{\infty} \beta_k x_k(t)$, may be truncated after a certain number of terms without significant loss of accuracy (Eubank 1999). Then choosing the first λ terms leads to an approximation of the true function. Taking into account this approximation, the model is now linear and of the form $y_i = f(t_i) + \epsilon_i$ where $f(t) \approx \sum_{k=1}^{\lambda} \beta_k x_k(t)$, with λ referred to as the smoothing parameter. As λ increases the model will better approximate the actual data. However, extremely large values of λ lead to a model that is nearly interpolating the data and as a result is less preferable since over-fitting the data increases the variability of the function to an undesirable level. Smaller values of this parameter lead to smoother functions but may not provide as accurate an estimate of the response due to the fact that the function will not be accurately capturing the nature of the data. Therefore, this simplification is not without a price, an estimate of λ that is in some sense an optimal value must now be found in order to continue.

3 SELECTING THE SMOOTHING PARAMETER

Though there are multiple methods for selecting the smoothing parameter, a popular choice is a variation of the cross validation technique. To facilitate the explanation of cross validation let us adopt the following notation: let $\mathbf{t}$ and $\mathbf{y}$ be vectors containing the independent and dependent data respectively, and let $\mathbf{b}$ be a vector containing the estimates of the actual coefficients, β, for the model. Also, since only the first λ terms of the series are used there will be λ known functions with which to build the estimator. Therefore $\mathbf{b}$ is of length λ and there is an $n \times \lambda$ matrix, $\mathbf{X}_\lambda$, that contains the λ components of the predictor function evaluated at each of the n points. Then $\mathbf{X}_\lambda$ is defined explicitly by $\{\mathbf{X}_\lambda\}_{i,j} = x_j(t_i)$, where $1 \leq i \leq n$ and $1 \leq j \leq \lambda$. Thus the estimate of $f(\cdot)$ can now be written in matrix form as $\hat{f} = \mathbf{X}_\lambda \mathbf{b}_\lambda$. To determine a value of λ that is optimal one must first have a method of computing $\mathbf{b}_\lambda$. This is simple enough since, for a given λ, the model is linear and thus a least-squares estimate of $\mathbf{b}_\lambda$ is given by $\mathbf{b}_\lambda = (\mathbf{X}_\lambda^T \mathbf{X}_\lambda)^{-1} \mathbf{X}_\lambda^T \mathbf{y}$.

Once the least-squares estimate of $\mathbf{b}_\lambda$ is available the process of determining an optimal value of λ can begin. First, create n subsets of size $n-1$ where the i^{th} subset is generated by removing the point (t_i, y_i). Then let $\mathbf{x}_{\lambda i}$ be the vector of values obtained by evaluating each of the λ components of the prediction function at t_i and $\mathbf{b}_{\lambda(i)}$ be the set of estimated coefficients when the i^{th} point is removed. Next, calculate the n values of $\mathbf{b}_{\lambda(i)}$ corresponding to the n subsets. Once this is done let $\mu_{\lambda(i)}$ represent the estimated response, $\mathbf{x}_{\lambda i}^T \mathbf{b}_{\lambda(i)}$. Then the cross validation function is defined as

$$CV(\lambda) = n^{-1} \sum_{i=1}^{n} (y_i - \mu_{\lambda(i)})^2.$$

The idea is to generate an estimate of the prediction error with the known values that were held out of the previous computations. The method chosen here was a variation of cross validation known as generalized cross validation or GCV. The value of λ chosen from this process is the minimizer of the GCV criterion; that is the value of λ for which GCV(λ) is the smallest.

There are certain problems that arise when generalized cross validation is utilized. First, there is the problem of which minimum to choose. A global minimum is guaranteed but there is also the possibility of several local minima. Choosing the first local minimum is not advisable since it generally leads to choices of λ than are smaller than desired for multiple reasons. Smaller values of λ tend to generate a function that estimates the data poorly in addition to providing confidence bands with lower coverage probabilities. However, the global minimum is not always the optimal choice if the set, Λ, that λ is chosen from is allowed to remain unbounded. This case tends to provide values that are too large and thus lead to near interpolation of the data. The solution then is to experimentally determine an appropriate set Λ from which to choose λ as well as investigating the advantages from using the global minimum in each scenario.

4 VARIANCE ESTIMATION

Regardless of the selection criteria used for λ, once an optimal value has been chosen the next step is to then find an estimate of variance for the data at hand. Several methods for variance estimation are available. Since the model is linear it seems reasonable to employ the variance estimator from linear regression

$$\sigma_\lambda^2 = \frac{\sum_{j=1}^{n}\left(y_j - \sum_{k=1}^{\lambda} c_k x_k(t_j)\right)^2}{(n-\lambda)} = \frac{RSS(\lambda)}{(n-\lambda)}$$

where $RSS(\lambda)$ is simply the residual sum of squares associated with the estimate of $\mathbf{b}_\lambda$. However, it is important to note that the optimal value of λ for determining the model is not necessarily the optimal value for estimation of σ^2. Alternative methods that do not require a value of the smoothing parameter be chosen are also viable options. One such model that looks at the squared difference between successive terms was proposed by Rice (1984) and has the form

$$\hat{\sigma}^2 = (n-1)^{-1}\sum_{j=2}^{n}(y_j - y_{j-1})^2.$$

Another model with similar structure was introduced in Hall, Kay and Titterington (1990) and is given by

$$\hat{\sigma}^2 = (n-2)^{-1}\sum_{j=2}^{n-1}(.809y_{j-1} - .5y_j - .309y_{j+1})^2.$$

Of the three estimators, the first performed the best in the scenarios in which it was tested. It appears that both the second and third methods require a relatively dense grid in order to perform well. However, the estimator ultimately chosen was a more conservative estimate of the form $\frac{RSS(\lambda)}{(n-\lambda-1)}$. This had the benefit of underestimating the variance with less frequency than any of the other methods, but still estimating the variance accurately.

5 CONSTRUCTING THE BANDS

To proceed with constructing the confidence bands recall that $\hat{f}(t) = \mathbf{b}^T\mathbf{x}(t)$ and assume $Y \sim N_n(\mu, \sigma^2\mathbf{I})$. Then this implies that

$$\mathbf{b} = (\mathbf{X}_\lambda^T\mathbf{X}_\lambda)^{-1}\mathbf{X}_\lambda^T\mathbf{Y} \sim N_\lambda(b, \sigma^2(\mathbf{X}_\lambda^T\mathbf{X}_\lambda)^{-1})$$

since for $Z \sim N_n(\mu, \mathbf{\Sigma})$ it is known that $\mathbf{B}Z \sim N_k(\mathbf{B}\mu, \mathbf{B\Sigma B}^T)$ for $\mathbf{B}$ a $k \times n$ matrix see, for example, Moser (1996). The bias is due to the estimation of β, an infinite set of parameters, with $\mathbf{b}$, a finite set. Also, let $\mathbf{P}$ be a $\lambda \times \lambda$ matrix such that $\sigma^2(\mathbf{X}_\lambda^T\mathbf{X}_\lambda)^{-1} = \mathbf{P}^T\mathbf{P}$. Then consider a band of the form $\hat{f}(t) \pm c\hat{\sigma}p(t)$ where $p(t) = ||\mathbf{P}x(t)||$. Several choices of c are possible, here we take $c = [kF_{k,\nu;\ \alpha}]^{1/2}$, a Scheffé critical value where $k = \lambda$ and $\nu = n - \lambda - 1$ (Naiman 1986).

6 SIMULATION STUDIES

Confidence bands of the aforementioned form were applied to various functions for which both $n = 50$ and $n = 100$ design points were used. In two of the scenarios, cases one and three, $n = 200$ points were also used. The design points were obtained by using a uniform design scheme where $t_i = \frac{(b-a)(8i-4}{8n}$. In order to determine the Scheffé critical value a significance level of 95% was chosen and thus $\alpha = .05$. To get an estimate of the coverage probability resulting from such confidence bands a particular function was chosen and the proposed method was applied for two thousand simulated cases. The coverage probability is determined by dividing the number of times the estimated function value falls within the confidence bands by the number of simulations. For each scenario a plot of the confidence bands, actual function, and predicted function was investigated in order to visualize the coverage obtained. Also calculated for each scenario is the mean and standard deviation for both the smoothing parameter, λ, and the estimate of the function's standard deviation, $\hat{\sigma}$. The functions themselves were chosen in order to simulate data whose true nature may be difficult to determine without the actual function being known. For each scenario the interval was $[0, 20]$ so that the functions could exhibit some non-linear behavior and begin to level off in order to determine the effectiveness of the confidence bands.

6.1 Case 1

The first case corresponds to choosing $f(t) = \frac{15t}{t^2+1}$ with the true standard deviation of ϵ set at $\sigma = 1$. For this scenario Table 1 provides the means and standard deviations of both λ and $\hat{\sigma}$. Also shown in Table 1 are the coverage probabilities for the entire interval $[0, 20]$ and for a truncated interval of $[3, 20]$ given by p and p' respectively. A factor contributing to the difference between the coverage on the two intervals can be seen in Figure 1. The peak occurring at the left hand side of the interval causes difficulties in obtaining an accurate confidence band. When the boundary bias due to the cosine functions appearing in the estimator is also taken in account, the variation in the two probabilities is not surprising. Figure 2 shows a similar picture with the number of design points doubled. There is an obvious improvement between the trials as can be seen due to the tightening of the confidence bands. The result is higher coverage probabilities, particularly for the entire interval, that are likely due to the fact that the bias is approaching zero as the sample size is increased. However, since the confidence bands are somewhat conservative even with 100 design points there is not a clear conclusion. Even as can be seen in Figure 2 a line with a slope between zero and negative one could easily be fit without straying outside the confidence bands, thus not accounting for the peak. By increasing to $n = 200$ the confidence bands fit the data closer and thus make a linear relationship of slope zero seem less likely, as can be seen in Figure 3.

n	p	p'	$\overline{x}_\lambda$	s_λ	$\overline{x}_{\hat{\sigma}}$	$s_{\hat{\sigma}}$
50	.8775	.9525	10.78	3.86	1.14	0.12
100	.9755	.9950	13.60	2.64	1.08	0.08
200	.9805	.9825	15.11	2.28	1.04	0.05

Table 1: Results from Case 1.

Figure 1: Interval estimates, estimated function, and actual function for Case 1 with $n = 50$ design points.

Figure 2: Interval estimates, estimated function, and actual function for Case 1 with $n = 100$ design points.

Figure 3: Interval estimates, estimated function, and actual function for Case 1 with $n = 200$ design points.

6.2 Case 2

The second scenario is a horizontal shift of the first function given by $f(t) = \frac{15(t-10)}{(t-10)^2+1}$ with the actual standard deviation remaining one. Since this shift centers the root in the interval $[0, 20]$ the effect of the boundary bias is reduced. Thus Table 2 shows the coverage probability for the entire interval, p, as well as the mean and standard deviation for both the distributions of λ and $\hat{\sigma}$. Figure 4 shows the upper and lower confidence bands as well as the actual function and estimated function for $n = 50$. Figure 5 shows a similar picture for $n = 100$. Here there seemed to be no reason to determine bands when $n = 200$ as the bands for both 50 and 100 captured the nature of the date extremely well.

n	p	$\overline{x}_\lambda$	s_λ	$\overline{x}_{\hat{\sigma}}$	$s_{\hat{\sigma}}$
50	.9985	17.82	1.92	1.26	0.12
100	.9985	19.08	1.23	1.12	0.08

Table 2: Results from Case 2.

Figure 4: Interval estimates, estimated function, and actual function for Case 2 with $n = 50$.

Figure 5: Interval estimates, estimated function, and actual function for Case 2 for $n = 100$.

6.3 Case 3

The third case study was for a rational function as well. In this case the function was $f(t) = \frac{t^4-42t^3+630t^2-4023t+9315}{t^4-36t^3+486t^2-2916t+6642}$ with an actual standard deviation of $\sigma = 1/2$. Table 3 shows the coverage probability for the entire interval, p, as well as again displaying the mean and standard deviation for both the distributions of λ and $\hat{\sigma}$. Figure 6 also depicts the upper and lower confidence bands as well as the actual function and estimated function while Figure 7 shows the outcome for $n = 100$ design points. Again the confidence bands are wide enough to easily allow a linear function of slope zero to fit for $n = 50$ and only tighten slightly when $n = 100$. Thus it is again beneficial to look at the results for $n = 200$. As can be seen in Figure 8 the confidence bands have become narrow enough to support the presence of a relationship in the data.

n	p	$\overline{x}_\lambda$	s_λ	$\overline{x}_{\hat{\sigma}}$	$s_{\hat{\sigma}}$
50	.8740	10.32	2.99	0.57	0.06
100	.9605	11.54	2.18	0.53	0.04
200	.9825	12.29	2.05	0.52	0.03

Table 3: Results from Case 3.

Figure 6: Interval estimates, estimated function, and actual function for Case 3, $n = 50$

Figure 7: Interval estimates, estimated function, and actual function for Case 3 with $n = 100$.

Figure 8: Interval estimates, estimated function, and actual function for Case 3 for $n = 200$.

6.4 Case 4

The last scenario corresponds to choosing

$$f(t) = \frac{50}{\sqrt{2\pi \cdot (2.5)^2}} e^{-\frac{(t-10)^2}{2 \cdot 2.5^2}}.$$

For this choice of $f(t)$ the actual error standard deviation was again chosen to be $\sigma = 1/2$. The normal function was scaled by a factor of 50 in order to make $\sigma = 1/2$ a reasonable choice. The values of 10 for the mean and 2.5 for the standard deviation were chosen to center the function in the interval and to ensure that four standard deviations fall within the interval $[0, 20]$. The following table then gives the coverage probability for the entire interval, p, as well as the mean and standard deviation for both the distributions of λ and $\hat{\sigma}$. Figure 9 shows both confidence bands as well as the actual and estimated functions. Figure 10 shows the results obtained by doubling the number of design points while Figure **??** shows the confidence bands for $n = 200$ design points.

n	p	$\overline{x}_\lambda$	s_λ	$\overline{x}_{\hat{\sigma}}$	$s_{\hat{\sigma}}$
50	.9450	9.28	2.40	0.56	0.06
100	.9295	9.59	1.98	0.53	0.04
200	.9790	9.90	1.83	0.51	0.03

Table 4: Results from Case 4.

Figure 9: Interval estimates, estimated function, and actual function for Case 4 when $n = 50$.

Figure 10: Interval estimates, estimated function, and actual function for Case 4 for $n = 100$.

Figure 11: Interval estimates, estimated function, and actual function for Case 4 for $n = 200$.

7 Conclusion

As a result of the simulation studies it can be seen that the sample size, n, affects the construction of confidence bands as well as the results garnered from them. Increasing the sample size induces a reduction in the bias and thus an increase in coverage probabilities. The importance of this idea is that the confidence bands, in practice, would be used to make inferences about the data at hand. A major component of that is verifying the shape of the data; that is to say, determine whether the peak seen in the data is a spurious anomaly that can be attributed to random error or that the data is actually related by some specific, inherent pattern. The results show there is a greater ability to do just this for larger values of n.

There are multiple areas for future research in the application of such confidence banding techniques. In particular would be the possibility of confidence bands used with spline estimation or with longitudinal data. Another topic of interest is relaxing the normality condition on the error terms and investigate the performance of these procedures asymptotically.

In addition it should be noted the the procedure presented is relatively simple to implement. It only requires obtaining a variance estimate, factorization and multiplication of matrices, and utilization of an F-distribution. Since the entirety of this process is carried out in a least squares setting, any software package such as SAS can be easily used to implement the method without a great expense of resources.

APPENDIX

A. Sample Program

The following code can be used to generate the results for the scenario described by the first case in the simulation studies with 200 design points.

```
program basic
    use imsl
    implicit none
    double precision :: pi, a, b, c
    integer :: n, lambda, i, j, k, L, index, count, count2,
    simnumber, total, total2, num, num2
    common / comblock / pi, a, b, n
    double precision, allocatable, dimension(:) :: e, t, y, mu, fhat,
    lowest, upest
    double precision, allocatable, dimension(:) :: lowavg, upavg,
    fhatavg, gavg
    double precision :: sigma=1,trace, g, f, RSS, small, SigEst, q0,
    q1, ratio, ratio2
    double precision, allocatable, dimension(:,:) :: GCV, Id, temp,
    blambda, Xlambda
    double precision, allocatable, dimension(:,:) :: Slambda, M, var,
    Pmat, p
    pi = 4*datan(dble(1.0))
    n = 200; lambda=0; RSS=0; small=0; SigEst=0; L=0; index=0; L=0
    i=0; j=0; k=1
    a = 0; b = 20; simnumber=1000; ratio=0; ratio2=0; num=0;
    num2=0

    call rnset(0)

    allocate(lowavg(n), upavg(n), fhatavg(n), gavg(n))

do k=1,simnumber


allocate(Id(n,n))

allocate(e(n),t(n),y(n),mu(n),fhat(n),p(n,1),upest(n),lowest(n))
```

```
!create a random set of iid normal errors and a grid over the
!interval 0 to 1.

call drnnoa(n,e)

do i=1, n
    t(i) = (b-a)*(8*dble(i)-4.0)/(8*dble(n))
end do

!scale the error vector and define function y

e=e*sigma

do i=1, n
    y(i) = g(t(i)) + e(i)
end do



allocate(GCV(20,1))
do L=1, 20

    allocate (Xlambda(n,L))
        call buildXlam(L, t, Xlambda)
    allocate (blambda(L,1))
    allocate (Slambda(n,n))
    allocate (M(L,n))
        call matrix(L, y, Xlambda, Slambda, blambda, mu, M)
    call identity(Id,n)
    allocate(temp(n,n))
    temp = Id-Slambda
    RSS = sum((y-mu)**2)
    call tr(temp,n,trace)
    GCV(L,1) = RSS/((trace/n)**2)/n
deallocate (Xlambda, blambda, Slambda, M, temp)
end do
```

```
!Now choose smallest value from GCV loop to determine lambda

small = GCV(1,1)
index = 1
do L=1, 19
    if (  GCV(L+1,1) < small  ) then
        small = GCV(L+1,1)
        index = L
    end if
end do

!Reallocate and define matrices with optimal lambda

lambda = index

allocate (Xlambda(n,lambda))

allocate (blambda(lambda,1))

allocate (Slambda(n,n))

allocate (M(lambda,n))

allocate (var(lambda,lambda))

allocate (Pmat(lambda,lambda))



call buildXlam(lambda, t, Xlambda)

call matrix(lambda, y, Xlambda, Slambda, blambda, mu, M)

!Estimate sigma

call standard(y,t,lambda,SigEst)

!Bulid variance matrix, var

call varmat(lambda, M, SigEst, var, Pmat)
```

```
!build fhat

call band(t, lambda, blambda, fhat, p, Pmat, SigEst, lowest,
upest, c)

!Check accuracy of point-wise band estimates

call check(count, count2, upest, lowest, t, total, total2)


call printer(lambda, blambda, GCV, mu, y, SigEst, t)


do i=1,n

lowavg(i)  = lowavg(i)  +  lowest(i)

upavg(i)   = upavg(i)   + upest(i)

fhatavg(i) = fhatavg(i) + fhat(i)

gavg(i)    = gavg(i) + g(t(i))

end do

num  = num  + count
num2 = num2 + count2

deallocate(M, var, Pmat, e, t, y, mu, fhat, lowest, upest, GCV,
Id, blambda, Xlambda, p, Slambda)

end do

ratio  = dble(dble(num)/dble(simnumber))

ratio2 =dble(dble(num2)/dble(simnumber))

lowavg = lowavg/simnumber

upavg = upavg/simnumber

fhatavg =fhatavg/simnumber

gavg = gavg/simnumber
```

```
call printer2(ratio, ratio2, simnumber, lowavg, upavg, gavg,
fhatavg, lambda, SigEst)

end program basic




!!!!!!!!!!!!!!!!!!!!!Functions and Subroutines!!!!!!!!!!!!!!!!!!

double precision function g(t)
    implicit none

    double precision, intent (IN) :: t
    double precision :: pi, a, b
    integer :: n
    common / comblock / pi, a, b, n

    g = 15.0*t/(t**2+1.0)

end function



double precision function q0(t)
    implicit none

    double precision, intent (IN) :: t

    q0 = t-(t**2)/2

end function



double precision function q1(t)
    implicit none

    double precision, intent(IN) :: t

    q1 = (t**2)/2

end function
```

```
double precision function f(t,j)
    implicit none

    double precision, intent (IN) :: t
    integer, intent (IN) :: j
    double precision :: pi, a, b, q0, q1
    integer :: n
    common / comblock / pi, a, b, n

    if (j==1) then
            f=1
        else if (j==2) then
            f = q0(t)
        else if (j==3) then
            f = q1(t)
        else
            f=dsqrt(dble(2.0/(b-a)))*dcos(dble(j-3)*t*pi/(b-a))
    end if

end function



subroutine buildXlam(lambda, t, Xlambda)
    implicit none

    double precision :: pi, a, b, f
    integer :: n
    common / comblock / pi, a, b, n
    integer, intent(IN) :: lambda
    double precision,dimension(n,lambda),intent(OUT) :: Xlambda
    double precision, intent(IN), dimension(n) ::  t
    integer :: i,j

do i=1, n
    do j=1, lambda
        Xlambda(i,j) = f(t(i), j)
    end do
end do

end subroutine buildXlam
```

```
subroutine matrix(lambda, y, Xlambda, Slambda, blambda, mu, M)
    implicit none

    double precision :: pi, a, b
    integer :: n
    common / comblock / pi, a, b, n
    integer, intent(IN):: lambda
    double precision, allocatable, dimension(:,:) :: XtransX,
    transINV
    double precision, allocatable, dimension(:,:) :: temp
    double precision, intent(IN), dimension(n,lambda) :: Xlambda
    double precision, intent(OUT), dimension(lambda,1) ::
    blambda
    double precision, intent(IN), dimension(n,1) :: y
    double precision, intent(OUT), dimension(n,n) :: Slambda
    double precision, intent(OUT), dimension(n,1) :: m
    double precision, intent(OUT), dimension(lambda,n) :: M


!Build Xlambda transpose times Xlambda called XtransX and invert
!since it is needed to build Slambda and blambda
    allocate (XtransX(lambda,lambda))
        XtransX = matmul(transpose(Xlambda),Xlambda)
    allocate (transINV(lambda,lambda))
        call dlinrg(lambda, XtransX, lambda, transINV, lambda)


!Build Slambda
    allocate (temp(lambda,n))
    temp = matmul(transINV, transpose(Xlambda))
    Slambda = matmul(Xlambda, temp)
    deallocate(temp)


!Build matrix, M, with which to compute var, the variance matrix
!for the distribution of the b lambdas.
    M = matmul(transINV, transpose(Xlambda))


!Build blambda
    blambda = matmul(M, y)
```

```
!Build mu
    mu = matmul(Slambda, y)
    deallocate(transINV, XtransX)

end subroutine matrix



subroutine varmat(lambda, M, SigEst, var, Pmat)
    implicit none

    double precision :: pi, a, b
    integer :: n, i, j
    common / comblock / pi, a, b, n
    integer, intent(IN):: lambda
    double precision, intent(IN), dimension(lambda,n) :: M
    double precision, intent(IN) :: SigEst
    double precision, intent(OUT), dimension(lambda,lambda) ::
    var
    double precision, intent(OUT), dimension(lambda,lambda) ::
    Pmat
    double precision, dimension(lambda,lambda) :: temp
    logical :: pivot=.FALSE.
    integer, dimension(n) :: piv

    var = SigEst*matmul(M, transpose(M))\\
    call dlchrg(lambda, var, lambda, pivot, piv, temp, lambda)

    do i=1,lambda
        do j=1,lambda
            if (i>j) then
                Pmat(i,j) = 0
            else
                Pmat(i,j) = temp(i,j)
            end if
        end do
    end do

end subroutine varmat
```

```
subroutine tr(matrix,dim,trace)
    implicit none

    integer ::i
    integer, intent(IN) :: dim
    double precision, intent(IN),dimension (dim,dim) :: matrix
    double precision, intent(OUT) :: trace
    trace = 0

!Define the trace of a a matrix
    do i=1,dim
        trace = trace + matrix(i,i)
    end do

end subroutine tr



subroutine identity(Id,dim)
    implicit none

    integer :: i, j
    integer, intent(IN) :: dim
    double precision, intent(OUT), dimension (dim,dim) :: Id

!Build dim by dim identity matrix
    do i=1,dim
        do j=1,dim
            if (i==j) then
                Id(i,j) = 1
            else
                Id(i,j) = 0
            end if
        end do
    end do

end subroutine identity



subroutine standard(y,t,lambda,SigEst)
    implicit none

    double precision :: pi, a, b, g, temp
    integer :: n, i, j
    common / comblock / pi, a, b, n
    integer, intent(IN):: lambda
    double precision, intent(IN), dimension(n) :: y, t
    double precision, intent(OUT) :: SigEst

temp = 0

do i=1,n
    temp = temp+(y(i)-g(t(i)))**2
end do

SigEst = dsqrt(temp/(n-lambda-1))

end subroutine standard

subroutine band(t,lambda,blambda,fhat, p, Pmat, SigEst, lowest,
upest, c)
    implicit none

    double precision :: pi, a, b, dfin, SigEst, f, temp
    integer :: n, i, j
    common / comblock / pi, a, b, n
    integer, intent(IN)  :: lambda
    double precision, dimension(lambda,1) :: x
    double precision, intent(IN), dimension(n) :: t
    double precision, intent(IN), dimension(lambda,1) :: blambda
    double precision, intent(OUT), dimension(n) :: fhat, upest,
    lowest
    double precision, intent(OUT), dimension(n,1) :: p
    double precision, intent(IN), dimension(lambda,lambda) ::
    Pmat
    double precision, dimension(1,1) :: temp2
    double precision, intent(OUT) :: c

    temp = 0

    do j=1,n
        do i=1,lambda
            temp = temp + blambda(i,1)*f(t(j),i)
        end do
        fhat(j) = temp
        temp = 0
    end do
    do i=1,n
      do j=1,lambda
        x(j,1) = f(t(i),j)
      end do

    temp2 =
    dsqrt(matmul(transpose(x),matmul(transpose(Pmat),matmul(Pmat,x))))
    p(i,1) = temp2(1,1)
    temp2 = 0

    end do

    c =
    dsqrt(dble(lambda)*dfin(dble(.95),dble(lambda),dble(n-lambda)))

    do i=1,n
        upest(i) = fhat(i) + c*SigEst*p(i,1)
        lowest(i) = fhat(i) - c*SigEst*p(i,1)
    end do

end subroutine band


subroutine check(count, count2, upest, lowest, t, total, total2)
    implicit none

    double precision :: pi, a, b, g
    integer :: i, j, n, total, total2
    common / comblock / pi, a, b, n
    integer, intent(OUT) :: count, count2
    double precision, intent(IN), dimension(n) :: upest, lowest,
    t

    count=1; count2=1
    i = 1; j = 7

     do while (count==1 .AND. i<=50)
        if (  lowest(i) > g(t(i))  .OR.  upest(i) < g(t(i))  )
        then
            count = 0
        end if
        i = i + 1
        total = total + count
    end do

    do while (count2==1 .AND. j<=50)
        if (  lowest(j) > g(t(j))  .OR.  upest(j) < g(t(j))  )
        then
            count2 = 0
        end if
        j = j + 1
        total2 = total2 + count2
    end do

end subroutine check

subroutine printer(lambda, blambda, GCV, mu, y, SigEst, t)
    implicit none

    double precision :: pi, a, b, SigEst
    integer :: n, i, j
    common / comblock / pi, a, b, n
    integer, intent(IN) :: lambda
    double precision, intent(IN), dimension(n) :: mu, y, t
    double precision, intent(IN), dimension(lambda,1) :: blambda
    double precision, intent(IN), dimension(20,1) :: GCV

open(0,file='superfile.dat', position='append')
    write(0,*) lambda, SigEst
close(0)

open(1, file='blambda.dat', status='replace')
    do j=1,lambda
        write(1,*) blambda(j,1)
    end do
close(1)

open(2,file='GCV.dat', status = 'replace')
    do j=1, 20
        write(2,*) j,  GCV(j,1)
    end do
close(2)

open(3,file='data.dat',status='replace')
    write(3,*) "        t(i)                    mu(i)
    y(i)"
    do j=1, n
        write(3,*) t(j), mu(j), y(j)
    end do
close(3)

end subroutine printer

subroutine printer2 (ratio, ratio2, simnumber, lowavg, upavg,
gavg, fhatavg, lambda, SigEst)
    implicit none

    double precision :: pi, a, b
    integer :: n, i
    common / comblock / pi, a, b, n
    double precision, intent(IN) :: ratio, ratio2, SigEst
    double precision, intent(IN), dimension(n) :: lowavg, upavg,
    gavg, fhatavg
    integer, intent(IN) :: simnumber, lambda

open(4,file='average.dat',status='replace')
    write(4,*) "       lowavg                  fhatavg
    upavg"
    do i=1,n
        write(4,*) lowavg(i), fhatavg(i), upavg(i)
    end do
close(4)

open(5,file='final.dat', position='append')
    write(5,100) ratio,ratio2,lambda,SigEst
    100 format("r=",f6.4"  r2=",f6.4,"  lambda=",i2,"
    sigma=",f12.8)
close(5)

end subroutine printer2
```

REFERENCES

Eubank, Randall L.(1999), *Nonparametric Regression and Spline Smoothing*, New York: Marcel Dekker, Inc.

Hall, P., Kay, J., and Titterington, D. M. (1990), "Asymptotically Optimal Difference based Estimation of Variance in Nonparametric Regression," *Biometrika*, 77, 521-528.

Moser, Barry Kurt (1996), *Linear Models: A Mean Model Approach*, San Diego: Academic Press.

Naiman, Daniel Q. (1986), "Conservative Confidence Bands in Curvilinear Regression," *The Annals of Statistics*, 14, 896-906.

Rice, J. (1984), "Bandwidth Choice for Nonparametric Regression," *The Annals of Statistics*, 12, 1215-1230.

Scheffé, Henry (1953), "A Method for Judging all Contrasts in the Analysis of Variance," *Biometrika*, 40, 87-104.

Sun, Jiayang, and Loader, Clive R. (1994), "Simultaneous Confidence Bands for Linear Regression and Smoothing," *The Annals of Statistics*, 22, 1328-1345.

Wynn, Henry P., and Bloomfield, P. (1971), "Simultaneous Confidence Bands in Regression Analysis," *The Journal of the Royal Statistical Society. Series B*, 33, 202-217.