

The Volume-of-Tube formula: Computational Methods and Statistical Applications

Catherine Loader*
Department of Statistics
Case Western Reserve University
Cleveland, OH 44106

July 5, 2004

Abstract

The volume-of-tube formula was first introduced by [Hotelling \(1939\)](#), to solve significance testing problems in nonlinear regression. Following this pioneering work, significant research has been performed on extending the tube formula to more general settings, including multidimensional problems. Applications of these results in statistical inference include confidence bands in regression and smoothing models; functional data analysis; testing in mixture models; and spatial scan analysis.

Implementation of the tube formula requires numerical evaluation of certain problem-specific geometric constants that appear in Hotelling's formula and its extensions. The purpose of this article is to describe a software library, `libtube`, that performs the calculations. Illustrative applications are given.

1 Introduction

The volume-of-tube problem can be stated rather simply. Given a curve (or manifold) \mathcal{M} lying in n -dimensional Euclidean space, what is the volume of the set of all points lying within a radius r of the curve? In statistical applications, the spherical version of this problem often arises; the manifold lies on the surface of the unit sphere in n dimensions, and one wishes to compute the $(n - 1)$ -dimensional volume (or surface area) of the set of points lying within a distance r of the manifold.

[Hotelling \(1939\)](#) formulated and solved the volume-of-tube formula, motivated by application to significance testing in nonparametric regression. A companion paper, [Weyl \(1939\)](#), extended the results to higher dimensional manifolds; that is, when \mathcal{M} is a surface, or more generally when \mathcal{M} is a manifold of dimension $d \leq n$. More recent developments of the tube formula and related methods, and statistical applications, can be found in [Knowles and Siegmund \(1989\)](#), [Johansen and Johnstone \(1990\)](#), [Naiman \(1990\)](#), [Sun and Loader \(1994\)](#), [Lin \(1997\)](#), [Takemura and Kuriki \(2002\)](#), [Taylor \(2002\)](#) and [Pilla and Loader \(2003\)](#).

The main purpose of this article is to describe a set of routines written by the author to implement the volume-of-tube formula in statistical problems. In [Section 2](#) the tube formula (with boundary corrections) is described. The

*Research supported by NSF grant DMS 0603202 and ONR grant N00014-04-1-0481.

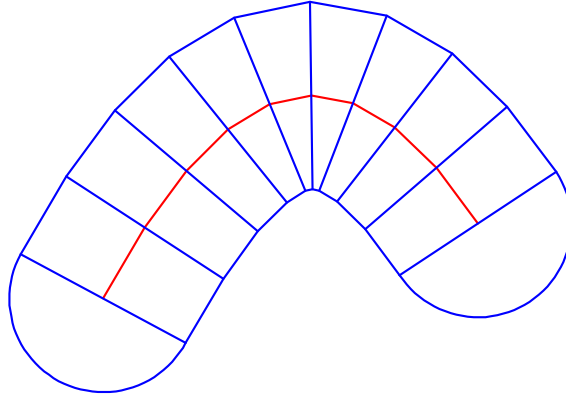


Figure 1: The manifold is represented by the red curve. The tubular neighborhood is approximated by trapezoids, plus the two end-point caps.

`libtube` software is described in Section 3. Applications to non-linear regression, simultaneous confidence bands and mixture modeling are described in Sections 4, 5 and 6 respectively.

2 The Volume-of-Tube Formula

The volume-of-tube formula was first derived by Hotelling (1939). The result can be illustrated on the plane by reference to Figure 1. The manifold is represented by the red curve. The tubular neighborhood of a given radius r is approximated by trapezoids, plus the two end-point caps. Adding up the area of the trapezoids and letting the partition become increasingly fine shows that the area (or two-dimensional volume) of the tube is

$$\text{Length of Manifold} \times 2r + \pi r^2.$$

The $2r$ represents the cross-sectional area of the manifold, while πr^2 represents the area of the end-point caps. The result extends to manifolds embedded in n -dimensional space;

$$\text{Volume} = \kappa_0 V_{n-1} r^{n-1} + \frac{l_0}{2} V_n r^n. \quad (1)$$

Here κ_0 is the length of the manifold and l_0 is the number of end-points (often, $l_0 = 2$). The functions $\psi_0(r)$ are the cross-sectional area and volume of the end-point caps respectively, and $V_k = \pi^{k/2}/\Gamma(1 + k/2)$ is the volume of the k -dimensional unit sphere.

The formula (1) is exact, provided that the radius is sufficiently small. It becomes approximate when the tube ‘overlaps’; in Figure 1, overlap occurs when the radius is large enough that the cross-section lines intersect each other.

When the manifold lies on the unit sphere, the result is similar, but the cross-sectional area is replaced by partial beta functions. The result is

$$\text{Volume} = \frac{\kappa_0 A_n}{2\pi} P(B_{1,(n-2)/2} \geq w^2) + \frac{l_0 A_n}{4} P(B_{1/2,(n-1)/2} \geq w^2),$$

where $B_{a,b}$ denotes a random variable following a beta distribution with parameters a and b ; $A_n = 2\pi^{n/2}/\Gamma(n/2)$ is the surface area of the unit sphere in R^n , and $w = 1 - r^2/2$.

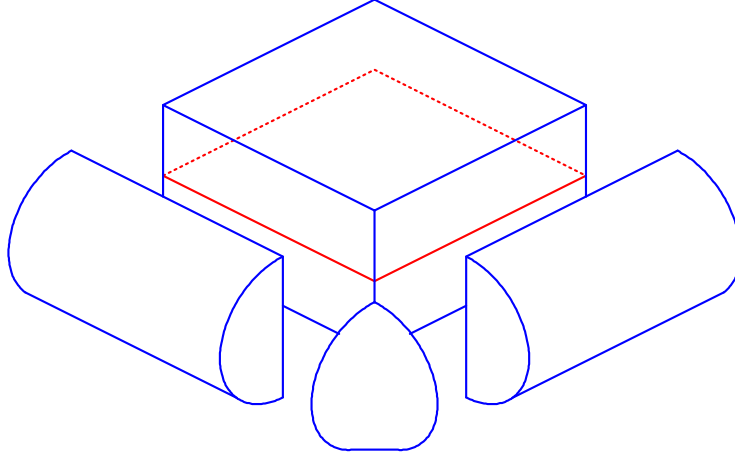


Figure 2: Tube around a two dimensional manifold. The manifold is shown in red, and the tube is divided into a main part, half-cylinders around the edges, and corner wedges.

2.1 Multidimensional Manifolds

Figure 2 shows a tube around a two-dimensional manifold. To compute the volume of the tubular neighborhood, one divides the tube into different pieces: a main piece, the half-cylinders around each edge, and wedges at each corner of the manifold. For higher dimensional manifolds, the ideas are similar, but there are more pieces to take care of.

Hotelling's results were extended to manifolds with $d \geq 2$ by [Weyl \(1939\)](#). [Naiman \(1990\)](#) provided boundary corrections. The result is a series with $d + 1$ terms. The first four terms (for manifolds on the unit sphere) are

$$\begin{aligned}
 \text{Volume} &= \frac{\kappa_0 A_n}{A_{d+1}} P(B_{(d+1)/2, (n-d-1)/2} > w^2) \\
 &+ \frac{l_0 A_n}{2A_d} P(B_{d/2, (n-d)/2} > w^2) \\
 &+ \frac{\kappa_2 + l_1 + m_0}{2\pi} \frac{A_n}{A_{d-1}} P(B_{(d-1)/2, (n-d+1)/2} > w^2) \\
 &+ \frac{l_2 + m_1 + n_0}{4\pi} \frac{A_n}{A_{d-2}} P(B_{(d-2)/2, (n-d+2)/2} > w^2). \quad (2)
 \end{aligned}$$

The constants l_0 , l_1 and l_2 arise from the corresponding series for the half-tubes around the boundaries of the manifold. l_0 is the $(d - 1)$ -dimensional volume of the boundaries (or the total length of the four edges in Figure 2). l_1 and l_2 are higher order terms representing boundary curvature.

The constants m_0 and m_1 arise from the 'corner wedges' where two boundary faces meet. In Figure 2, m_0 is the sum of the four wedge angles at each corner of the manifold.

The constant n_0 arises for manifolds with $d \geq 3$, from the corners where three (or more) boundary faces meet.

2.2 Random Processes

In statistical applications, the fundamental use of the tube formula is to find (or at least approximate) the distribution of the maximum of certain random processes. As an example, consider the process

$$Z(\lambda) = \langle T(\lambda), U \rangle$$

where $T(\lambda)$ is an R^n -valued vector function, U is uniformly distributed over the unit sphere and $\langle \cdot, \cdot \rangle$ denotes the usual inner product. Suppose that one is interested in finding

$$P(\sup_{\lambda} Z(\lambda) \geq w)$$

for some w .

The inner product exceeds w if, and only if, U is sufficiently close to $T(\lambda)$. Specifically,

$$\|T(\lambda) - U\|^2 = \|T(\lambda)\|^2 - 2\langle T(\lambda), U \rangle + \|U\|^2 = 2(1 - \langle T(\lambda), U \rangle).$$

Hence, $\|T(\lambda) - U\| \leq r$ if, and only if, $\langle T(\lambda), U \rangle \geq w$, where $r^2 = 2(1 - w)$. The probability (2.2) is therefore

$$\frac{\text{Area of tube of radius } r \text{ around } \{T(\lambda)\}}{\text{Surface area of unit sphere in } R^n}.$$

In many statistical applications, one is interested in the distribution of the maximum of a Gaussian process,

$$Z(\lambda) = \langle T(\lambda), \epsilon \rangle$$

where ϵ follows the standard multivariate normal distribution. To reduce this to the uniform process, one needs to condition on the length of the ϵ vector, and integrate over the conditional distribution; see [Sun and Loader \(1994\)](#). The final result, up to fourth order, is

$$\begin{aligned} P(\sup_{\lambda} Z(\lambda) \geq c) &\approx \frac{\kappa_0}{A_{d+1}} P(\chi_{(d+1)/2}^2 \geq c^2) \\ &+ \frac{l_0}{2A_d} P(\chi_{d/2}^2 > c^2) \\ &+ \frac{\kappa_2 + l_1 + m_0}{2\pi A_{d-1}} P(\chi_{(d-1)/2}^2 > c^2) \\ &+ \frac{l_2 + m_1 + n_0}{4\pi A_{d-2}} P(\chi_{(d-2)/2}^2 > c^2), \end{aligned} \quad (3)$$

where χ_k^2 denotes a chi-square random variable with k degrees of freedom.

3 The libtube Library

The main computational problem in implementing results based on the tube formula is evaluation of the constants κ_0 , κ_2 , l_0 e.t.c. The `libtube` library implements the tube library up to fourth order terms. To use the library, one must first write a ‘manifold function’ defining the problem. The `libtube` functions take the manifold function as input, and use numerical integration method to compute the constants. The present implementation evaluates the terms up to fourth order as in (2), so this is a complete implementation of the tube formula for manifolds with $d \leq 4$.

Source code for the library can be downloaded from <http://www.herine.net/stat/tube/>. The library is written in C, and can be compiled on Linux systems using

```
% make
% make install
```

This will compile two libraries: the main `libtube` library, and a mathematical utility library `libmut`. The latter includes routines for such tasks as numerical integration and linear algebra. The libraries are installed to `/usr/local/lib`.

To use the libraries, one must write a main calling function and manifold function. If these are in a file named, say, `nlreg.c`, the command to compile is

```
% cc -o nlreg nlreg.c -ltube -lm
```

The library (and the examples given in this paper) have been written and tested using the Gnu C compiler available in most Linux distributions. The C code should be compatible with most other compilers and operating systems.

3.1 Function Calls

The top-level functions provided by `libtube` are:

- `tube_constants()`, to numerically evaluate κ_0 and the other constants appearing in (2).
- `tailp()` and `critval()`, which compute tail probabilities corresponding to a specified cut-off, and critical values corresponding to a specified significance level.

3.1.1 Calling sequence for `tube_constants()`

The function to compute the constants is

```
int tube_constants(f,d,m,ev,mg,fl,kap,wk,terms,uc)
double *fl, *kap, *wk;
int d, m, ev, *mg, (*f)(), terms, uc;
```

The arguments to this function are:

- `f` The manifold function to compute $l(x)$ and its derivatives.
- `d` The dimension of the manifold.
- `m` The maximum length of the $l(x)$ vectors. The argument provided is only used to allocate work space; the actual length of $l(x)$ is returned by the manifold function.
- `ev` Integration type. For rectangular domains, `ISIMPSON` is the most useful. `IMONTE` uses Monte-Carlo integration to evaluate κ_0 only. `ISPHERIC` can be used for circular and spherical domains in 2 and 3 dimensions. `IDERFREE` uses a derivative-free methods to evaluate κ_0 , when $d = 1$ only.
- `mg` Integer vector, giving the number of partitions to use in each dimension of the numerical integration rules.
- `fl` Integration limits. A numeric vector with length $2d$. The first d components give lower limits for each variable; the remaining d components give upper limits.
- `kap` is the vector through which the computed constants are returned. It should be allocated with at least $\min(d + 1, 4)$ terms. The values returned are $\kappa_0, l_0/2, (\kappa_2 + l_1 + m_0)/(2\pi)$ and $(l_2 + m_1 + n_0)/(4\pi)$.
- `wk` is a workspace vector. If `wk=NULL`, the required workspace will be allocated and freed within the `tube_constants()` function. To pre-allocate the space, the required length can be found by calling `k0_reqd(d,m)`.

- **terms** Number of terms to compute, from 1 to 4. Specifying **terms=2** results in only κ_0 and l_0 being computed. This never requires second derivatives of the manifold, but has the cost of reduced accuracy when $d \geq 2$.
- **uc** An indicator variable indicating whether the manifold function computes the weight vectors **uc=0** or covariance derivative matrix **uc=1**.

The value returned by `tube_constants()` is the number of terms actually computed. It will be equal to $\min(d + 1, \text{terms}, 4)$.

3.1.2 Calling sequence for `tailp()` and `critval()`

Tail probabilities are computed using the function `tailp`:

```
double tailp(c,k0,m,d,s,n,process)
double c, *k0, n;
int m, d, s, process;
```

Critical values corresponding to a specified tail probability are computed using the `critval` function:

```
double critval(alpha,k0,m,d,s,n,process)
double *k0, al, n;
int m, d, it, s;
```

These arguments represent:

- **c** Cut-off value for `tailp()`.
- **alpha** tail probability for `critval()`.
- **k0** is the vector of constants computed by the `tube_constants()` function.
- **m** is the number of terms provided by the **k0** vector. This is the value returned by the `tube_constants()` function.
- **d** is the dimension of the manifold.
- **s** Either `ONE_SIDED` or `TWO_SIDED`.
- **n** For the t-process, the residual degrees of freedom used to estimate σ . For the uniform process, the dimension n . Ignored for the Gaussian process. Beware that n must have type double.
- **process** Either `GAUSS` (when ϵ is multivariate Gaussian); `TPROC` (Gaussian process with estimated variance); or `UNIF` (when ϵ is uniform on the unit sphere).

3.2 Manifold Functions

Suppose the manifold is defined by a vector function $T(x)$ mapping a d -dimensional domain \mathcal{X} to the manifold \mathcal{M} in n -dimensional space. The constants in the tube formula can be computed from $T(x)$ and its derivatives, so in its simplest form, the manifold function simply computes these. In statistical applications, one usually doesn't get $T(x)$ naturally, but rather one gets a vector $l(x)$ such that $T(x) = l(x)/\|l(x)\|$ (see the regression examples in Sections 4 and 5). The manifold function can instead provide $l(x)$ and its derivatives.

In still other examples, one doesn't even obtain $l(x)$ directly, but instead obtains a covariance function $\sigma(x, x') = \langle l(x), l(x') \rangle$ (see the mixture example, Section 6). Since the distance between any two points on the manifold is given by

$$\|l(x) - l(x')\|^2 = \sigma(x, x) + \sigma(x', x') - 2\sigma(x, x'),$$

knowledge of the covariance function determines $l(x)$ up to an orthogonal transformation. The manifold function can provide $\sigma(x, x')$ and its derivatives.

The precise form of the manifold functions is illustrated by the examples.

3.2.1 Writing a Manifold Function with vectors

The manifold function computes the vector $l(x)$ and its derivatives. The basic form of the function is

```
int mymf(x, l, reqd)
double *x, *l;
int reqd;
{ /* function body goes here */
}
```

The x argument is a point in the input space; l is a vector to be filled in by the manifold function. The final argument, $reqd$, is an integer indicating what the library requires from the manifold function. If $reqd=0$, only the vector $l(x)$ is required. If $reqd=1$, then both $l(x)$ and $l'(x)$ (or all the first-order partial derivative vectors of $l(x)$) are required. If $reqd=2$, then additionally the second-order partial derivative vectors are required.

In statistical applications, the manifold function will generally require a data vector, sample size n , dimension d and variables other than x in order to perform its calculations. These variables should be assigned to global variables so that they are accessible in the manifold function.

The results of the computations are returned through the l vector. The vector $l(x)$ is placed in the first n elements. The first-order derivatives are placed in the next $n \times d$ elements. The second-order derivatives are placed in the next $n \times d \times d$ elements.

The function should return n , the length of the vector $l(x)$. Generally, this should be equal to the n value provided in the `tube_constants()` call; it should never be larger. It can be less. An example where it may be less is for a kernel regression with compactly supported kernel; only the non-zero elements of $l(x)$ need be retained.

3.2.2 Writing a Manifold Function with a covariance function.

The structure of a manifold function based on the covariance is identical to the vector case; it differs in what is computed.

Given a covariance function $\sigma(x, x')$, the manifold function needs to compute (in the one-dimensional case),

$$\begin{pmatrix} \sigma(x, x') & \frac{\partial \sigma(x, x')}{\partial x'} & \frac{\partial^2 \sigma(x, x')}{\partial x'^2} \\ \frac{\partial \sigma(x, x')}{\partial x} & \frac{\partial^2 \sigma(x, x')}{\partial x \partial x'} & \frac{\partial^3 \sigma(x, x')}{\partial x \partial x'^2} \\ \frac{\partial^2 \sigma(x, x')}{\partial x^2} & \frac{\partial^3 \sigma(x, x')}{\partial x^2 \partial x'} & \frac{\partial^4 \sigma(x, x')}{\partial x^2 \partial x'^2} \end{pmatrix},$$

evaluated at $x' = x$. Again, the matrix is stored in the vector l , with the columns stacked atop each other.

In higher dimensions, the required matrix is most easily written in terms of differential operators. The required $(1 + d + d^2) \times (1 + d + d^2)$ matrix is

$$\begin{pmatrix} I \\ D_{x_1} \\ \vdots \\ D_{x_d} \\ D_{x_1, x_1} \\ \vdots \\ D_{x_d, x_d} \end{pmatrix} \sigma(x, x') \begin{pmatrix} I & D_{x'_1} & \dots & D_{x'_d} & D_{x'_1, x'_1} & \dots & D_{x'_d, x'_d} \end{pmatrix}$$

where D represents the partial derivative operator with respect to the subscripted variables.

Another view is as follows. If \mathbf{L} is the matrix computed by a manifold function with vectors, then $\mathbf{L}^T\mathbf{L}$ is the matrix computed by a manifold function with a covariance function.

4 Application to Nonlinear Regression

This was the motivating example considered by [Hotelling \(1939\)](#), and was developed in much more detail by [Knowles and Siegmund \(1989\)](#). Suppose one has data $(x_i, Y_i), i = 1, \dots, n$, and a nonlinear regression model, such as

$$Y_i = \alpha e^{\gamma x_i} + \epsilon_i. \quad (4)$$

The important feature of this model is that the parameter α enters the model linearly, while γ enters nonlinearly. Assume that the errors are independent $N(0, \sigma^2)$.

Consider the problem of testing $H_0 : \alpha = 0$ vs $H_1 : \alpha \neq 0$. It can be shown that the log-likelihood ratio test statistic is equivalent to

$$L = \frac{\min_{\alpha, \gamma} \|Y - a l(\gamma)\|^2}{\|Y\|^2} \quad (5)$$

where $l(\gamma)^T = (e^{\gamma x_1}, \dots, e^{\gamma x_n})$.

In classical statistical theory, log-likelihood ratio test statistics often have asymptotic χ^2 distributions. However, this is not the case for the statistic (5). One way to see this is to recall that proofs of the χ^2 results are based on a quadratic expansion of the statistic under the null parameters. For the present problem this would require an expansion around $(0, \gamma_0)$ where γ_0 is ‘the’ null value of γ . Unfortunately this is undefined: when $\alpha = 0$, the parameter γ does not appear in (4); it is not identifiable!

For fixed γ , minimizing over a is a linear least-squares problem. It follows that

$$L = 1 - \sup_{\gamma} \left\langle \frac{l(\gamma)}{\|l(\gamma)\|}, \frac{Y}{\|Y\|} \right\rangle^2.$$

The null hypothesis H_0 is rejected if $L \leq 1 - w^2$ for some $w > 0$, or equivalently, if

$$\sup_{\gamma} \left| \left\langle \frac{l(\gamma)}{\|l(\gamma)\|}, \frac{Y}{\|Y\|} \right\rangle \right| \geq w.$$

The constant w must be chosen to obtain a specified significance level. That is, we need to be able to evaluate probabilities of the form

$$P(\sup_{\gamma} |\langle T(\gamma), U \rangle| \geq w) \quad (6)$$

where $T(\gamma) = l(\gamma)/\|l(\gamma)\|$ defines a curve on the unit sphere, and $U = Y/\|Y\|$ is (under $H_0 : \alpha = 0$) uniformly distributed on the surface of the sphere.

Code implementing the tube formula for the non-linear regression problem is given in [Appendix A](#). The manifold function `regmf()` computes the components of $l(\gamma)$; $l_i(\gamma) = e^{\gamma x_i}$, and of $l'(\gamma)$, $l'_i(\gamma) = x_i e^{\gamma x_i}$. These vectors are stored end-to-end in the `l` argument.

The main routine `main()` first reads in the data. the data and computes the tube constants. Limits for γ are set at $(-4, 4)$ (to use $(-\infty, \infty)$, we would need to reparameterize the model). The `tube_constants()` function is then called, and the resulting constants (κ_0 and $l_0/2$) are printed out. The call to `critical()` computes the 5% critical value for the test statistic.

Running the program with the `nlreg.dat` data file produces the following output.

```

Data filename ? nlreg.dat
n = ? 10
kappa0 = 2.26367 10/2 = 1.00000
critical value: 0.72193

```

5 Application to Simultaneous Confidence Bands

Application of the tube formula to find simultaneous confidence bands for regression models has been studied in [Naiman \(1987\)](#), [Sun and Loader \(1994\)](#) among others. Consider again regression data, but now suppose that the model is

$$Y_i = a_0 + a_1x_i + a_2x_i^2 + \epsilon_i = \mu(x_i) + \epsilon_i$$

(although we formulate the problem for quadratic regression, extension to other linear models is straightforward). The goal is to find confidence bands

$$\hat{\mu}(x) \pm c\sqrt{\text{var}(\hat{\mu}(x))}$$

with simultaneous coverage over some nice domain \mathcal{X} :

$$P(|\hat{\mu}(x) - \mu(x)| \leq c\sigma\|l(x)\| \text{ for all } x \in \mathcal{X}) = 1 - \alpha. \quad (7)$$

The least-squares estimates of the parameters are

$$\begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \hat{a}_2 \end{pmatrix} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y$$

where \mathbf{X} is the design matrix. For fixed x , $\mu(x)$ is estimated by

$$\hat{\mu}(x) = \hat{a}_0 + \hat{a}_1x + \hat{a}_2x^2 = (1 \ x \ x^2) (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y = \langle l(x), Y \rangle,$$

and the variance of the estimate is $\text{var}(\hat{\mu}(x)) = \sigma^2\|l(x)\|^2$.

Now, $\hat{\mu}(x) - \mu(x) = \langle l(x), \epsilon \rangle$, and the probability (7) is equivalent to

$$\alpha = P\left(\sup_x \left| \left\langle \frac{l(x)}{\|l(x)\|}, \epsilon \right\rangle \right| > c\right).$$

This problem can be solved using the Gaussian process variant of the tube problem.

Suppose $\mathbf{X} = \mathbf{Q}\mathbf{R}$ is the QR -decomposition of the design matrix. Then $l(x)$ lies in the column space of \mathbf{Q} for all x , and so

$$Z(\gamma) = \left\langle \frac{\mathbf{Q}^T l(x)}{\|\mathbf{Q}^T l(x)\|}, \mathbf{Q}^T \epsilon \right\rangle,$$

so it suffices to work with the vector $l^*(x) = \mathbf{Q}^T l(x) = (\mathbf{R}^T)^{-1} f(x)$ where $f(x)$ is a vector of the polynomial basis functions. The derivatives are easily found;

$$\frac{d}{dx} l^*(x) = (\mathbf{R}^T)^{-1} \frac{d}{dx} f(x)$$

and so on.

Code for the quadratic regression computations, in an arbitrary number of dimensions, is given in [Appendix B](#). The functions `quad()`, `quadi()` and `quadij()` compute the quadratic basis functions $f(x)$; first-order partial derivatives and second-order partial derivatives respectively. The manifold function is `quadmf()`. The `main()` function reads the data from a file;

computes the design matrix and its QR-decomposition; and then calls the `tube_constants()` function (The QR functions, `qr()` and `qrinvx()`, as well as `transpose()`, are part of the `nut` library).

When the program is run, the user is prompted for a data file (containing a matrix of the predictor variables); data dimension (n and d), and limits for the confidence band computation.

The tube constants are computed, then the critical value c for 95% confidence bands. Note that the final argument to `critval` is the residual degrees of freedom used to estimate σ ; [Sun and Loader \(1994\)](#) give the modification of (3) for this case.

Running the program on the `qreg.dat` file provided with `libtube` gives the following results.

```
Data filename ? qreg.dat
n = ? 9
dim = ? 2
Limits, Variable 1: -2 2
Limits, Variable 2: -2 2
Constants: 10.66325  5.01995 -0.69711  0.00000
Level 0.05 critical value:  7.34308
```

6 Application to Mixture Models

Suppose X_1, \dots, X_n are an i.i.d. sample from a density

$$f_{\alpha, \mu}(x) = (1 - \alpha)f_0(x) + \alpha\phi(x, \mu)$$

where α and μ are unknown parameters, with $0 \leq \alpha \leq 1$. The object is to test $H_0 : \alpha = 0$ vs $H_1 : \alpha > 0$. This is an example of mixture testing: under H_0 , the single component $f_0(x)$ describes the data, while under H_1 , the two components are required. Application of the tube formula to mixture testing has been discussed in [Lin \(1997\)](#) and [Lin and Lindsay \(1997\)](#).

The score process proposed by [Pilla and Loader \(2003\)](#) is

$$S(\mu) = \sum_{i=1}^n \frac{\phi(X_i, \mu)}{f_0(X_i)} - 1.$$

Under the null hypothesis, this has mean 0 and covariance function $n\sigma(\mu, \mu^\dagger)$, where

$$\sigma(\mu, \mu^\dagger) = \int \frac{\phi(x, \mu)\phi(x, \mu^\dagger)}{f_0(x)} dx - 1.$$

The normalized score process is $S^*(\mu) = S(\mu)/\sqrt{n\sigma(\mu, \mu)}$. This asymptotically behaves like a Gaussian process $Z(\mu)$, with mean 0 under H_0 , and a nonzero mean under H_1 . The maximum of the normalized score process serves as the test statistic.

Since an explicit vector representation of $Z(\mu)$ is not readily available, the manifold function must be written using the covariance function and its partial derivatives.

There is one additional difficulty. The normalized score process has a singularity at $\mu = 0$. For this reason, the manifold function works with Taylor series expansions of the covariance in this region. Also, the singularity results in a discontinuity in $S^*(\mu)$, and the manifold therefore has two pieces, and $l_0 = 4$. The computations will not recognize the singularity, so the output has to be adjusted manually.

The program for the mixture test is given in [Appendix C](#). The manifold function is `mixmf()`. The main routine sets limits for μ as $(-3, 3)$, calls the

`tube_constants()` function, and computes the 5% critical value. Output of the program is:

```
kappa0 = 5.27449
10/2 = 2.00000
Level 0.05 critical value = 2.49455
```

References

- Hotelling, H. (1939). Tubes and spheres in n -spaces, and a class of statistical problems. *American Journal of Mathematics*, 61:440–460.
- Johansen, S. and Johnstone, I. (1990). Hotelling’s theorem on the volume of tubes: some illustrations in simultaneous inference and data analysis. *The Annals of Statistics*, 18:652–684.
- Knowles, M. and Siegmund, D. (1989). On Hotelling’s geometric approach to testing for a nonlinear parameter in regression. *International Statistical Review*, 57:205–220.
- Lin, Y. (1997). *The Likelihood Ratio Test of Mixture Hypothesis and the Tube Volume Problem*. PhD thesis, Department of Statistics, The Pennsylvania State University.
- Lin, Y. and Lindsay, B. G. (1997). Projections on cones, chi-bar squared distributions, and weyl’s formula. *Statistics and Probability Letters*, 32:367–376.
- Naiman, D. Q. (1987). Simultaneous confidence bounds in multiple regression using predictor variable constraints. *Journal of the American Statistical Association*, 82:214–219.
- Naiman, D. Q. (1990). On volumes of tubular neighborhoods of spherical polyhedra and statistical inference. *The Annals of Statistics*, 18:685–716.
- Pilla, R. S. and Loader, C. (2003). The volume-of-tube formula: Perturbation tests, mixture models and scan statistics. Unpublished Manuscript.
- Sun, J. and Loader, C. (1994). Simultaneous confidence bands for linear regression and smoothing. *The Annals of Statistics*, 22:1328–1345.
- Takemura, A. and Kuriki, S. (2002). On the equivalence of the tube and euler characteristic methods for the distribution of the maximum of gaussian fields over piecewise smooth domains. *The Annals of Applied Probability*, 12:768–796.
- Taylor, J. (2002). Gaussian volumes of tubes, euler characteristic densities and correlated conjunctions. In *Proceedings of “Singular models and geometric methods in Statistics”*, pages 7–33. Institute of Statistical Mathematics.
- Weyl, H. (1939). On the volume of tubes. *American Journal of Mathematics*, 61:461–472.

A Nonlinear Regression: Source Code

```
#include <stdio.h>
#include <math.h>
#include <tube.h>
#define MAXN 1000

double x[MAXN], y[MAXN];
int n;

int regmf(gam,l,reqd)
double *gam, *l;
int reqd;
{ int i;
  double *l1;
  l1 = &l[n];
  for (i=0; i<n; i++)
  { l[i] = exp(gam[0]*x[i]);
    l1[i] = x[i]*exp(gam[0]*x[i]);
  }
  return(n);
}

int main()
{ FILE *infile;
  char filename[100];
  int i, t, mg;
  double gamlimits[2], kappa[4], c;
  printf("Data filename ? "); scanf("%s",filename);
  printf("n = ? "); scanf("%d",&n);
  infile = fopen(filename,"r");
  for (i=0; i<n; i++) fscanf(infile,"%lf%lf",&x[i],&y[i]);
  gamlimits[0] = -4.0;
  gamlimits[1] = 4.0;
  mg = 200;

  t = tube_constants(regmf,1,n,ISIMPSON,&mg,gamlimits,kappa,NULL,2,0);
  printf("kappa0 = %8.5f    10/2 = %8.5f\n",kappa[0],kappa[1]);

  c = critval(0.05,kappa,t,1,TWO_SIDED,(double)n,UNIF);
  printf("critical value: %8.5f\n",c);
}
```

B Simultaneous Confidence Bands: Source Code

```
#include <stdio.h>
#include <tube.h>
#include <mutil.h>

int dim, n, p;
double *X;

void quad(x,f)
double *x, *f;
{ int i, j, k;
  k = 0;
  f[k++] = 1.0;
  for (i=0; i<dim; i++) f[k++] = x[i];
  for (i=0; i<dim; i++)
    for (j=i; j<dim; j++)
      f[k++] = x[i]*x[j];
}

void quadi(x,f,i0)
double *x, *f;
int i0;
{ int i, j, k;
  k = 0;
  f[k++] = 0.0;
  for (i=0; i<dim; i++) f[k++] = (i==i0);
  for (i=0; i<dim; i++)
    for (j=i; j<dim; j++)
      f[k++] = (i==i0)*x[j] + (j==i0)*x[i];
}

void quadij(x,f,i0,j0)
double *x, *f;
int i0, j0;
{ int i, j, k;
  k = 0;
  f[k++] = 0.0;
  for (i=0; i<dim; i++) f[k++] = 0.0;
  for (i=0; i<dim; i++)
    for (j=i; j<dim; j++)
      f[k++] = ((i==i0) & (j==j0)) + ((i==j0) & (j==i0));
}

int quadmf(x,l,reqd)
double *x, *l;
int reqd;
{ int i, j, k;
  k = 0;
  quad(x,l);
  qrtinvx(X,l,n,p);
  k++;
  for (i=0; i<dim; i++)
    { quadi(x,&l[k*p],i);
```

```

    qrtinvx(X,&l[k*p],n,p);
    k++;
}
for (i=0; i<dim; i++)
    for (j=0; j<dim; j++)
        { quadij(x,&l[k*p],i,j);
          qrtinvx(X,&l[k*p],n,p);
          k++;
        }
return(p);
}

int main()
{ FILE *infile;
  char filename[100];
  int i, j, mg[100], t;
  double xlim[100], kappa[4], datarow[100], c;
  printf("Data filename ? "); scanf("%s",filename);
  printf("n = ? "); scanf("%d",&n);
  printf("dim = ? "); scanf("%d",&dim);
  infile = fopen(filename,"r");
  if (infile==NULL)
  { printf("Error: can't read input file\n");
    return(0);
  }
  p = 1 + dim + dim*(dim+1)/2;
  X = (double *)calloc(n*p,sizeof(double));
  for (i=0; i<n; i++)
  { for (j=0; j<dim; j++)
    fscanf(infile,"%lf",&datarow[j]);
    quad(datarow,&X[i*p]);
  }
  transpose(X,n,p);
  qr(X,n,p,NULL);
  for (i=0; i<dim; i++) mg[i] = 20;
  for (i=0; i<dim; i++)
  { printf("Limits, Variable %1d: ",i+1);
    scanf("%lf%lf",&xlim[i],&xlim[i+dim]);
  }
  t = tube_constants(quadmf,dim,p,ISIMPSON,mg,xlim,kappa,NULL,4,0);
  printf("Constants: %8.5f %8.5f %8.5f %8.5f\n",
    kappa[0],kappa[1],kappa[2],kappa[3]);
  c = critval(0.05,kappa,t,dim,TWO_SIDED,(double)(n-p),TPROC);
  printf("Level 0.05 critical value: %8.5f\n",c);
}

```

C Mixture Example: Source Code

```
#include <stdio.h>
#include <math.h>
#include <tube.h>
#define MAXN 1000

double x[MAXN], y[MAXN];
int n;

int mixmf(mu,l,reqd)
double *mu, *l;
int reqd;
{ double emm, mm;

  if (fabs(mu[0]) < 0.01)
  { mm = mu[0]*mu[0];
    l[0] = 1 + mm/2*(1 + mm/3*(1 + mm/4*(1 + mm/5)));
    l[1] = 0.5*(1 + mm/3*(2 + mm/4*(3 + mm/5*(4 + 5/6*mm)));
    l[1] = l[2] = mu[0]*l[1];
    l[3] = 0.5*(1 + mm/3*(4 + mm/4*(9 + mm/5*(16+25/6*mm)));
  } else
  { emm = exp(mu[0]*mu[0]);
    l[0] = emm-1;
    l[1] = l[2] = mu[0]*emm;
    l[3] = emm*(1+mu[0]*mu[0]);
  }
  return(2);
}

int main()
{ int d, i, mg, t;
  double mulimits[2], kappa[4], c;
  mulimits[0] = -3.0;
  mulimits[1] = 3.0;
  mg = 200;
  d = 1;

  t = tube_constants(mixmf,d,n,ISIMPSON,&mg,mulimits,kappa,NULL,2,1);
  /* modify kappa[1] = 10/2 for the singularity */
  kappa[1] += 1.0;

  printf("kappa0 = %8.5f\n",kappa[0]);
  printf(" 10/2 = %8.5f\n",kappa[1]);
  c = critval(0.05,kappa,t,d,ONE_SIDED,0.0,GAUSS);
  printf("Level 0.05 critical value = %8.5f\n",c);
}
```