Representing Penalties on Basis Expansions as a Polynomial In the Parameters (Working Title)

April 2, 2013

1 Basis Function Expansion Methods

Fitting sophisticated mathematical functions to empirical data continues to be a challenge in statistical science. One approach, known as the basis function expansion method offers a considerable degree of flexibility and mathematically tractable solutions. This approach involves representing an unknown function $y(t)$ using a finite number of basis functions $\{\phi_k(t)\}$ through the conditional expectation

$$\mathbb{E}[y|t] = c_0\phi_0(t) + c_1\phi_1(t) + \cdots + c_N\phi_N(t),$$

where $\{c_k\}$ is a set of appropriately chosen coefficients.

Here, $\mathbb{E}[y|t]$ is the conditional expectation of the dependent variable y, given our independent data x. The conditional expectation $\mathbb{E}[y|t]$ is the choice of y depending exclusively on t that minimises the squared loss $\mathbb{E}[y(t) - E[y|t]]^2$. This is a theoretical optimiser, and so in practice we must make do with an estimate $\widehat{\mathbb{E}[y|t]}$ instead, using our basis functions $\{\phi_k\}$

Example 1. (Simple Linear Regression) Simple linear regression is an example of a statistical procedure that uses a basis function representation. Take two basis functions $\{1, t\}$, so that $\hat{y}(t)$, our estimate of y given the value t, can be written as a linear combination of the two $\widehat{\mathbb{E}[y|t]} = \hat{y}(t) = c_0 + c_1 t$. This is the form of a simple linear regression model. If our basis consists of just the single constant function $\phi(t) = 1$ then we get a model of the form $\hat{y} = c_0$. In this case we would generally use the mean of the y values as our estimate of c_0. If we go in the other direction and add a quadratic function t^2 we get a quadratic regression model $\hat{y}(t) = c_0 + c_1 t + c_2 t^2$.

□

Example 2. (Broken Stick Function) Another choice of basis is the two functions $\{\phi_1, \phi_2\}$ defined as follows

$$\begin{aligned} \phi_1(t) &= 1 \\ \phi_2(t) &= \begin{cases} 0 & \text{if } t < \tau \\ t - \tau & \text{if } t \geq \tau. \end{cases} \end{aligned}$$

The first function is a constant function, the second is a ramp function based at τ.

This basis would work very well for representing data which starts to increase linearly past a certain threshold. Broken stick functions and constant functions are very cheap computationally, only needing literally a line or two of code to implement and only have at most one associated parameter (τ), consuming little memory. A computer can produce and work with comparatively a very large number of them at once. This could be used to cover situations where there data is unchanging over relatively long periods.

Ramp functions are the building blocks for the "P-splines" developed by Eiler and Marx. P-splines are a precursor to the penalty based approach used here. P-splines penalise the coefficients, whereas the focus here is to penalise the generated curve for some operator of it.

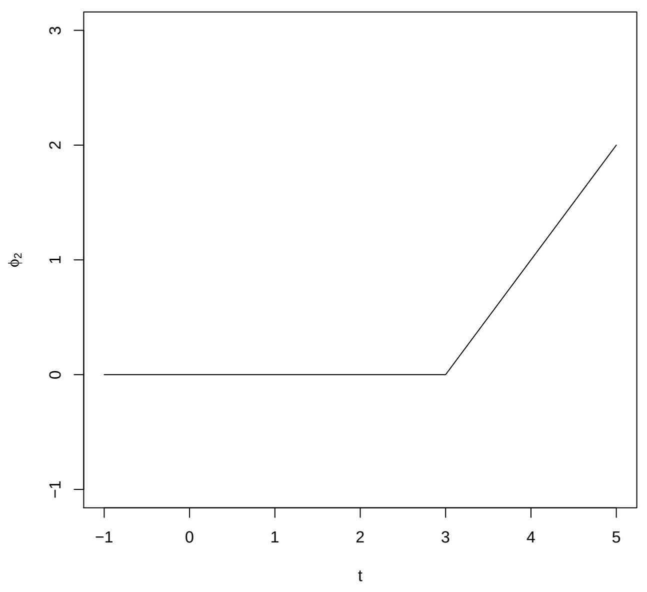

Figure 1: The broken stick function ϕ_2.

The above basis has two disadvantages though. Firstly the second basis function is only continuous, not differentiable. It would not be wise to use this basis if we wanted to estimate $\mathrm{d}y/\mathrm{d}t$. Secondly it is very arbitrary. It is not obvious why it would be useful compared to a quadratic model for example; the latter also allows to us estimate derivatives of all orders and can still accommodate data which varies in its rate of change.

□

This underscores an important point. There are many types of basis besides the ones usually encountered, such as polynomial bases. Many of them are fungible from the pure mathematical point of view in terms of how well they can fit a function. The statistician should make a sensible choice of basis. If our choice of basis is good, then it will be able to fit the data with only a few terms, and we might be able to avoid estimating many coefficients. This is one of the reasons why simple linear regression and quadratic regression are useful; they can capture much variation in the observed data in spite of being very simple.

Example 3. (Fourier Basis Functions) If our data has a periodic component to it, such as the observed temperature over the course of the year, or a time series of annual sales, then it seems intuitively correct to use a basis consisting of periodic functions. This suggests that we should use a Fourier basis consisting of the set of functions $\{\cos(n\omega t), \sin(n\omega t)\}$ where $0 \leq n \leq N$ for some N and ω is the frequency. The frequency depends on our time scale, they are related by the formula $\omega = 2\pi/T$, where T is the period.

Furthermore, Fourier basis functions have several other desirable properties. They are smooth, meaning that they can be used to estimate any derivative of the data, at least in theory. They are orthogonal, which can make certain problems more convenient and they are closed under differentiation, meaning that the derivative of a combination of Fourier basis functions, is itself a combination of Fourier basis functions. The latter two properties will prove very useful later.

A Fourier basis can represent any square integrable function on some interval. This covers a large proportion of the functions encountered in real life They cannot detect how frequencies change in space however, only their global behaviour.

□

Example 4. (B-spline Basis Functions) Roughly speaking, B-splines are compactly supported polynomial functions, or more practically they are nonzero only inside of a given interval. More formally a B-Spline basis consists of a degree n, which determines the degree of the basis functions, and a set of knots, that is a set of K time points $\{t_0, \ldots, t_K\}$.

Since they are compact they generally can only individually capture local information about the data. One of the main advantages of B Splines is that they can represent any other spline of the same degree and smoothness with the same knots. This makes them useful for statistics since make they fewer assumptions about the form of the data than other families of splines. For example the so-called natural cubic splines require that the second derivative of the fitted spline curve be zero at the boundary of the domain on which it is supported. If this was not true of the processes generating the data, then this could be a source of bias.

□

2 Least Squares Fitting

Least squares fitting is a means of fitting a function to data. The least squares fit is the one that minimises the sums of the squared errors. In the cases where we have n observations y_i measured at times t_i and where we are selecting our estimated function $\hat{y}(t)$ from a set of functions S, the least squares fit $\hat{y}_{LS}(t)$ is defined as

$$\hat{y}_{LS}(t) = \underset{f \in S}{\operatorname{argmin}} \sum_{i=1}^{n} [y_i - f(t_i)]^2.$$

From now on we will always be working with the sums of the squared errors, so we will not bother using any subscripts to indicate this and only use $\hat{y}(t)$ rather than $\hat{y}_{LS}(t)$.

Assume S can be spanned by some set of m basis functions i.e. $S = \{\sum_{i=1}^{m} c_i \phi_i(t) | c_i \in \mathbb{R}\}$. This suggests that to find $\hat{y}(t)$ it is only necessary to estimate to coefficients $\hat{c}_i$, so that $\hat{y}(t)$ never appears explicitly. Then we have completely determined $\hat{y}$. We can then write the estimate of the vector of coefficients $\hat{\mathbf{c}} = (\hat{c}_1, \ldots, \hat{c}_m)$ as:

$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathbb{R}^m}{\operatorname{argmin}} \sum_{i=1}^{n} [y_i - \sum_{j=1}^{m} c_j \phi_j(t_i)]^2.$$

The above expression is can be expressed more cleanly using vector notation. Firstly, we have $f(t) = \mathbf{c}'\phi(\mathbf{t})$, where $\mathbf{c} = (c_1, ..., c_m)$ and $\phi = (\phi_1(t), \ldots, \phi_2(t))'$. By constructing a matrix $\mathbf{\Phi}$, where the ith row of $\mathbf{\Phi}$ is $\phi(t_i)$, and let $\mathbf{y} = (y_1, \ldots, y_n)$ the least squares problem can written as

$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathbb{R}^m}{\operatorname{argmin}} (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}).$$

The expression on the right hand side is an example of a quadratic form, they are the generalisation of quadratic functions to finite dimensional vector spaces. We can have quadratic forms on infinite dimensional vectors spaces too, but that is not relevant here.

The advantage of quadratic forms is that they are very easy to minimise, one can use the normal equation, or compute the gradient and use an algorithim (or class of algorithms) such as conjugate gradient, quasi Newton methods or the Levenberg-Marquardt method.

3 Roughness Penalties

If we assume that at a given time t that, $\mathbf{y}(t) = \mathbf{\Phi}(t)\mathbf{c} + \mathbf{e}(\mathbf{t})$ where the elements of $\mathbf{e}$ are i.i.d. and have well defined second moments, thee least squares gives us the best linear unbiased estimator (BLUE) of $\mathbf{y}$, the function we assume to be generating the data. Nonetheless it is often useful to employ a form of regularisation which constrains how much $\hat{y}$ is allowed to vary. Intuitively, this reduces the variation of $\hat{y}$ and helps guard against overfitting. In the context of basis function expansions, the most commonly used penalty is the curvature penalty

$$\text{PEN}(\hat{y}) = \int_D [\hat{y}''(t)]^2 \mathrm{d}t.$$

Here D is our domain of interest and $\hat{y}''(t)$ stands for the second derivative of the function $\hat{y}(t)$.

This penalty tries to require $\hat{y}(t)$ to satisfy the differential equation $\hat{y}''(t) = 0$. The general solution of this ODE is of the form $\hat{y}(t) = \alpha + \beta t$, a straight line. The penalty has the effect of nudging $\hat{y}(t)$ towards linear regression.

The penalty can also be represented as a polynomial. Let $\langle f, g \rangle = \int_D f(t)g(t)\mathrm{d}t$. Here, $\langle \cdot, \cdot \rangle$ is the inner product on $L^2(D)$, the set of square integrable functions on D. The penalty can be written in the form $\text{PEN}(f) = \langle f'', f'' \rangle$. Substituting in the expansion for f we get

$$\begin{aligned}
\langle f'', f'' \rangle &= \langle \sum_{i=1}^{m} c_i \phi_i'', \sum_{i=1}^{m} c_i \phi_i'' \rangle, \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} \langle c_i \phi_i'', c_j \phi_j'' \rangle, \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j \langle \phi_i'', \phi_j'' \rangle, \\
&= \mathbf{c}'\mathbf{K}\mathbf{c}, \text{where } \mathbf{c} = (c_1, \ldots, c_m) \text{ is a vector}
\end{aligned}$$

Here, $\langle \phi_i'', \phi_j'' \rangle$ depends only on our choice of basis, since this generally doesn't change in the middle of an analysis, we can treat these terms as "fixed" for our purposes. In the last line the penalty term was represented as a quadratic form by defining an $m \times m$ matrix $\mathbf{K}$ by $\mathbf{K}_{ij} = \langle \phi_i'', \phi_j'' \rangle$ it can be seen that $\langle f'', f'' \rangle = \mathbf{c}'\mathbf{K}\mathbf{c}$.

The linearity property of the differentiation operator to expand out the inner product term, this technique is very useful for problems such as these.

4 Penalised Least Squares

When fitting curves to data, there are two distinct, but competing objectives. We want a fit $\hat{y}$ that maintains fidelity to the data, as represented by the goodness of fit, but simultaneously adheres to the requirement of a smooth description of the data.

The solution is to use a penalty that incorporates both of these objectives, the penalised sum of squared errors. The penalised sum of squared errors is a functional, or a function whose domain is a set of functions and whose codomain is the real numbers. Here it is a functional on the spanning set of our basis functions $\{\phi_i(t)\}$. It is the weighted sum of the roughness and least squares penalties

$$\text{PENSSE}(f) = \sum_{i=0}^{n} [y_i - f(t_i)]^2 + \lambda \int_D [f''(t)]^2 \mathrm{d}t.$$

The parameter λ controls the tradeoff between error and smoothness as represented by the least squares and roughness terms respectively. It can be thought of as a model complexity parameter; as λ increases $\hat{y}(t)$ becomes more linear and so our model tends more towards simple linear regression, as it decreases the model tends more towards fitting the data exactly.

In polynomial regression the analogous parameter would be the order of the polynomial we are fitting. However the two quantities differ in that the order of a polynomial is a discrete quantity, whilst λ is a continuous one.

We can also think of λ as controlling how closely the fitted curve adheres to the ODE $\hat{y}''(t) = 0$.

The choice of λ is important; strictly speaking we ought to denote $\hat{y}(t)$ as $\hat{y}_\lambda(t)$ to denote the dependence. An estimate for λ can generally be found by cross validation. As discussed in Green and Silverman sometimes there are better choices of λ than a computer can find though.

Assume again that $\hat{y}(t)$ is the sum of basis functions, depending on a vector of coefficients $\hat{\mathbf{c}}$. This means we can omit the dependence on $\hat{y}(t)$ and convert from the problem of estimating a function to estimating a vector $\hat{\mathbf{c}}(\lambda)$ (or just simply $\hat{\mathbf{c}}$) which depends on our choice of smoothing parameter. Using the previous results we can show that the penalised sum of squares can be written as the sum of two quadratic forms:

$$\text{PENSSE}(c) = (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}) + \lambda \mathbf{c}'\mathbf{K}\mathbf{c}.$$

Where $\mathbf{y}$, $\mathbf{K}$ and $\mathbf{\Phi}$ have the same definitions as they did before. This is an regularised regression problem or Tikhonov regularisation. The problem of minimising this expression has the solution

$$\begin{aligned}\hat{\mathbf{c}}(\lambda) &= (\mathbf{\Phi}'\mathbf{\Phi} + \lambda\mathbf{K})^{-1}\mathbf{\Phi}'\mathbf{y} \\ &= \mathbf{M}(\lambda)\mathbf{y}.\end{aligned}$$

Where we assume the inverse $\mathbf{M}(\lambda)$ exists.

If this problem is looked at from the perspective of linear models, as in Christessen, then $\mathbf{y} = \mathbf{\Phi c} + \mathbf{e}$, where the elements of $\mathbf{e}$ are i.i.d and have mean zero and variance σ^2, then we can estimate the variance of $\hat{y}(t)$. We will denote the matrix $\mathbf{M}(\lambda)$ by $\mathbf{M}$. The matrix $\mathbf{M}$ can be thought of as a sort of generalised projection matrix, since $\mathbf{M}(0)$ is the projection matrix on to the columns of $\mathbf{\Phi}$.

We have $\hat{\mathbf{c}} = \mathbf{M}(\mathbf{\Phi c} + \mathbf{e})$, so $\mathbb{E}[\hat{\mathbf{c}}] = \mathbf{H\Phi c}$. The variance of $\hat{\mathbf{c}}$ is given by $\text{Var}[\hat{\mathbf{c}}] = \sigma^2\mathbf{M\Phi\Phi}'\mathbf{M}'$. Hence the expected value of $\hat{\mathbf{y}}$ is $\mathbf{\Phi My}$ and its variance is $\sigma^2\mathbf{\Phi M\Phi\Phi}'\mathbf{M}'\mathbf{\Phi}'$.

Under the standard theory of linear models, the least squares estimate $\mathbf{M}(0)\mathbf{y}$ is the BLUE of $\mathbf{c}$. The contrapostive of this is that any linear estimater, including those of the form $\mathbf{M}(\lambda)\mathbf{y}$, can only achieve better reduction in variance by incurring bias. Any unbiased linear estimtor or a linear biased estimator with higher variance than the least squares estimator will have a higher mean error than the least square estimator, and is not worth paying attention to. The only class of linear estimators that can hope to do better than the least squares estimator are those that incur some bias in the hopes of achieving a sufficient reduction in variance to reduce the total mean square error against the least squares estimator. The estimator $\mathbf{M}(\lambda)$ should belong to the category, or there is no point in using it, so it is biased by design.

5 Multivariate Bases

What if our data is spatial in nature? In this case our data will often be at a series of points $\mathbf{x}_i = (x_i, y_i); i = 1, \dots, n$. It is actually not too difficult to extend our results; it is almost as easy as replacing the t_i with $\mathbf{x_i}$.

As before we expand our function y as a basis function expansion

$$y(\mathbf{x}) = \sum_{i=1}^{m} c_i \phi(\mathbf{x}).$$

Notice that even though we are working in more than one dimension, our sum is still "one dimensional" in that it is taken over a single index. This is deliberate as it makes our life much easier.

Finding a least squares estimate is identical, so we will not cover it here.

Example 5. (Fitting the Displacement of a Membrane Using a Laplacian Penalty) Suppose we had measurements of protrusion of a plate or membrane and we wished to fit some functions to it. We would assume the surface is not moving, as would be the case for clingfilm over a bowl for example. We could use standard interpolation or least squares methods. However the mechanics of these objects are often governed by Partial Differential Equations (PDEs). We would like to incorporate this information somehow.

If the perturbation of the surface is small, then it can be shown that a model of the mechanics of surface is a PDE called the wave equation. Let $u(x, y, t)$ be the displacement of the membrane at position (x, y) at time t. Then u is assumed to satisfy the partial differential equation:

$$u_{tt} = \frac{\partial^2 u}{\partial t^2} = c^2 \Delta u.$$

Here $u_{tt} = \partial^2 u/\partial t^2$, c is a parameter known as the wave speed and Δ is a differential operator known as the Laplacian. In two dimensions with Cartesian coordinates the Laplacian is defined as

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

We are interested in the steady state model, where the membrane does not move so that $u_{tt} = 0$. Of course u now only depends on the position so $u = u(x, y)$. In this case the wave equation reduces to Laplace's equation

$$\Delta u = 0.$$

Since the wave equation is not completely valid and only an approximation, we should not expect the data adhere perfectly to Laplace's equation anymore than we expect data to have a perfectly linear relationship. For example the surface could be bending too much for the wave equation to hold to high degree of accuracy. It is hoped that some use can be gotten out of the model though by using it in penalised regression with respect to the penalty function

$$\text{PENSSE}(f) = \sum_{i=0}^{n} (u_i - f(\mathbf{x}_i))^2 + \lambda \int_\Omega |\Delta f(\mathbf{x})|^2 \mathbf{dx}.$$

As usual we use $\hat{u}(x, y)$ to denote the function that minimises the penalty above that is then spanning set of our basis; it is an estimate.

The interpretation of the penalty in this case is a little more subtle than before. Harmonic functions, as solutions to Laplace's equation are called, are not flat as if they were linear, though linear functions are clearly harmonic. In two dimensions, compare the laplancian penalty to the curvature penalty that also generalises the one dimensional roughness penalty $\int_D [(u_{xx})^2 + (u_{xy})^2 + (u_{yy})^2] \mathrm{d}x\mathrm{d}y$. This penalty attempts to constrain all second order derivatives to zero. There are many harmonic functions which can vary over their domain, e.g $f(x, y) = e^x \sin(y)$. The laplacian penalty has more "give" than such penalties on functions, so we should not expect it to necessarily encourage the fitted surface to approach a plane as λ increases.

Generally λ would be chosen by a cross-validation approach as was the case for the roughness penalty. It is interesting to speculate about the implications of the order of magnitude of λ that optimises the cross-validation score. It could be thought of as a proxy for the validity of Laplace's equation. Intuitively the higher λ is the more the Laplacian penalty is emphasised, and so the more likely the model is to be valid.

The computational aspects of finding $\hat{u}(x, y)$ are identical to the case with one dimensional penalised regression. We will be able to express the penalty as a quadratic form on the coefficients of $\hat{u}(x, y)$ and then derived a penalised regression problem as before. Define an inner product on our functions by

$$\langle f, g \rangle = \int_\Omega f(\mathbf{x}) g(\mathbf{x}) \mathbf{dx}.$$

Then the laplacian penalty can be written as

$$\int_{\Omega} |\Delta f(\mathbf{x})|^2 \mathbf{dx} = \langle \Delta f, \Delta f \rangle.$$

If $f = \sum c_i \phi_i$ then $\Delta f = \sum c_i \Delta \phi_i$. In the same manner as the previous roughness penalty, we can see that if f is a basis expansion then we can write the penalty as a polynomial in the coefficients

$$\langle \Delta f, \Delta f \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j \langle \Delta \phi_i, \Delta \phi_j \rangle.$$

This can be written as $\mathbf{c}'\mathbf{K}\mathbf{c}$, here $\mathbf{K}_{ij} = \langle \Delta \phi_i, \Delta \phi_j \rangle$.

We define the matrix $\mathbf{\Phi}$ as before with $\mathbf{\Phi}_{ij} = \phi_j(\mathbf{x}_i)$, and let $\mathbf{c} = (c_1, \ldots, c_m)$. We can then write the penalty in terms of the coefficients

$$PENSSE(c) = (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}) + \lambda \mathbf{c}'\mathbf{K}\mathbf{c}.$$

□

6 Non Linear Penalties

Here we discuss generalisations of the roughess penalty to penaltys on nonlinear operators applied to the function. The problem is finding nice expressions for penalies of the form $\int_D |Tf(t)|^2 \mathrm{d}t$, where T is not necessarily a linear operator anymore. Continuing with the case where we are working on the span of a finite set of basis functions, so an arbitary function f can be written as $f(t) = \mathbf{c}'\phi(t)$, it will be shown how in some cases that the penalty can be written in the form $\mathbf{F}(\mathbf{c})'\mathbf{K}\mathbf{F}(\mathbf{c})$ where $F(c)$ is a vector valued function of our parameter vector $\mathbf{c}$. In all the cases we have seen so far $\mathbf{F}(\cdot)$ was simply the identity function $\mathbf{F}(\mathbf{c}) = \mathbf{c}$. In the nonlinear case things can get more interesting as expected.

The trick is that if our function is of the form $f = \sum c_i \phi_i$ then some nonlinear operators will return something of the form $\sum b_i(\mathbf{c})\psi_i$ for some other functions $\{\psi_i\}$. The coefficients b_i depend on the original coefficients c_i, reflecting the fact the our function is uniquely defined by and represented by its coefficients. Since the nonlinear operator T has the property that it maps from one vector space to another, the span of the ϕ_i functions to the span of the ψ_i functions. The nonlinearity is reflected in the fact the coefficients in front of the ψ_i depend nonlinearly on the coefficients of the ϕ_i functions.

Since the image (or range) of the span of the ϕ_i under T is a vector subspace of the inner product space $L^2[D]$ it inherits the inner product of its parent. This is the property that we will use, it is easy to find an expression for the inner product on a finite vector space, it will always be something of the form $\mathbf{b}'\mathbf{A}\mathbf{b}$. Since the coefficients of the ψ_i depend of the coefficients of the ϕ this can be written as $\mathbf{F}(\mathbf{c})'\mathbf{A}\mathbf{F}(\mathbf{c})$. The matrix A is generally easy to find, it is identical to the ideas we used before.

Finding the matrix $\mathbf{A}$ is easy it depends only on the ψ_i and can be written in the form $\mathbf{A}_{ij} = \langle \psi_i, \psi_j \rangle$. The new basis $\{\psi_i\}$ is self evident in many cases. The hard part is finding the coefficients, or equivalently the function $\mathbf{F}(\mathbf{c})$.

Example 6. ($f'' = f^2$) This example is purely artificial and intended to illustrate how the expresssion for the penalty term can be derived; it is a simple nonlinear ODE that is amenable to this technique. In this case our penalty is

$$\int_D (f''(t) - f(t)^2)^2 \mathrm{d}t.$$

$$f''(t) = \sum_{i=0}^{m} c_i \phi_i(t)''$$

$$f(t)^2 = \sum_{i=0}^{m}\sum_{j=0}^{m} c_i c_j \phi_i(t)\phi_j(t).$$

Hence,

$$f''(t) - f(t)^2 = \sum_{i=0}^{m} c_i \phi(t)'' - \sum_{i=0}^{m}\sum_{j=0}^{m} c_i c_j \phi_i(t)\phi_j(t).$$

Notice the second term is a two dimensional, finite sum. We now need to find norm. As before we can write this as an inner product: $\langle f'' - f^2, f'' - f^2\rangle$. The above expression is in the form $a_1\phi_1'' + \cdots + a_m\phi_m'' + b_{11}\phi_1\phi_1 + \ldots b_{mm}\phi_m\phi_m$, so we can represent it as $\mathbf{F}(\mathbf{c})'\mathbf{A}\mathbf{F}(\mathbf{c})$. The form above is a little awkward to work with. We would like to have it vary with one index only, or combine the ϕ_i'' and the $\phi_i\phi_j$ together. We would define a function $\pi(n)$ that returns the appropriate function either some ϕ_i or $\phi_i\phi_j$ depending on the value. Defining such a function is tricky however. One function is as follows

$$\pi(k) = \begin{cases} \phi_k'' & \text{if } k \leq m \\ \phi_{(k-1)|m}\phi_{(k-1) \bmod m} & \text{if } k > m \end{cases}$$

Here $a|b$ is integer division, i.e. $a|b = \text{floor}(a/b)$.

If we define $\psi_k(t)$ to be $\pi_k(t)$ and define a similar function $\sigma(i)$ for the coefficients we can define $f'' - f^2$ we get

$$f'' - f^2 = \sum \sigma(i)\psi_i.$$

We now can express the penalty as a quadratic form

$$\int_D (f(t)'' - f(t)^2)^2 dt = \sigma(\mathbf{c})'\mathbf{K}\sigma(\mathbf{c}).$$

Here $\mathbf{K}_{ij} = \langle\psi_i, \psi_j\rangle$ and $\sigma(\mathbf{c}) = (\sigma(1), \ldots, \sigma(m+m^2)) = (c_1, \ldots, c_m, c_1c_1, \ldots, c_mc_m)$. The inner product is $\langle f, g\rangle = \int_D f(t)g(t)dt$. Since $\sigma(\mathbf{c})$ is a second order polynomial in the c_i, the penalty is actually as fourth order polynomial in the c_i.

We can write the Penalised Sum of Squared Errors in terms of the c_i

$$PENSSE(\mathbf{c}) = (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c}) + \lambda\sigma(\mathbf{c})'\mathbf{K}\sigma(\mathbf{c}).$$

Alternative Representation Of The Penalty By Splitting the Inner Product

It is difficult to deal with the indexing above. An alternative is to break the function into two parts $f = f' + f^2$ where $f' = \sum c_k\phi_k'$ and $f^2 = \sum c_kc_l\phi_k\phi_l$. Since $\langle f', f^2\rangle = \langle f^2, f'$. We can represent the penalty in the form of the sum of three parts, letting $\mathbf{b} = (c_1c_1, c_1c_2, \ldots)$

$$\langle f' + f^2, f' + f^2\rangle = \mathbf{c}'\mathbf{Kc} + 2\mathbf{c}'\mathbf{Lb} + \mathbf{b}'\mathbf{Mb}.$$

Here $\mathbf{K}_{ij} = \langle\phi_i', \phi_j'\rangle$. $\mathbf{L}$ and $\mathbf{M}$ similarly represent $\langle f', f^2\rangle$ and $\langle f^2, f^2\rangle$.

Using the Vec Operator to Represent the Penalty

Definition. The Vec operator applied to a matrix stacks all the matrix's columns on top of each other

We can represent the vector of products c_ic_j as $\text{Vec}\,\mathbf{cc}'$. Note that each term c_ic_j can appear twice if $i \neq j$. We must therefore scale any inner product matrices appropriately. We can write the penalty without messing with indices

$$\langle f' + f^2, f' + f^2\rangle = \mathbf{c}'\mathbf{Kc} + 2\mathbf{c}'\mathbf{L}\,\text{Vec}(\mathbf{cc}') + \text{Vec}(\mathbf{cc}')'\mathbf{M}\,\text{Vec}(\mathbf{cc}')$$

□

7 Systems Of Ordinary Differential Equations (ODES)

The approach generalises quite well to systems of differential equations. In contrast to the multivariate case above, where one output variable depended on multiple input variables, we now have a time series of vectors $\mathbf{x}_k$ depending only on time. We give each component of $\mathbf{x}(t)$ its own basis expansion, but keep the same basis functions. For this section we will concentrate on the two dimensional case, except where it is obvious that we are considering an arbitrary number of dimensions.

$$\begin{aligned}\mathbf{x}_i(t) &= \mathbf{c}_i'\phi(t)\\ &= c_{i1}\phi_i(t) + \cdots + c_{im}\phi_m(t).\end{aligned}$$

Least Squares Penalty

We will also need an inner product of some sort to define the least squares and roughness penalties. It seems reasonable to use the standard dot product $\mathbf{x}\cdot\mathbf{x}$ in place of $|x|^2$ for our penalties. For a two dimensional series $\mathbf{x} = (x, y)$ with coefficient vectors $\mathbf{b}$ and $\mathbf{c}$, our least squares penalty has the form

$$\begin{aligned}SSE &= \sum_{i=1}^{n}\{[x_i - \sum_{j=1}^{m} b_j\phi_j(t_i)]^2 + [y_i - \sum_{j=1}^{m} c_j\phi_j(t_i)]^2\}\\ &= (\mathbf{x} - \mathbf{\Phi b})'(\mathbf{x} - \mathbf{\Phi b}) + (\mathbf{y} - \mathbf{\Phi c})'(\mathbf{y} - \mathbf{\Phi c})\\ &= \|[\mathbf{x}\,\mathbf{y}] - \mathbf{\Phi}[\mathbf{b}\,\mathbf{c}]\|_F^2.\end{aligned}$$

$\|\cdot\|_F^2$ is the Frobenius Norm. For an $n \times m$ matrix it is defined in terms of the sums of the squares of its elements

$$\|A\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}^2.$$

Differential Equation Penalty

A system of differential equations has the form

$$\begin{aligned}\mathbf{x}' &= f(\mathbf{x}, t)\\ \mathbf{x}(t_0) &= \mathbf{x_0}\end{aligned}$$

We won't be too worried about making sure our penalties are in the standard form above though. It sometimes convenient to leave a higher order derivative on the left hand side instead of converting them to the standard form.

As usual we will be dealing with expressions of the form $\|T\mathbf{x}\|$, except $\mathbf{x}(t)$ is a vector valued function, or a curve. We will use the following L^2 inner product to induce our norm

$$\langle \mathbf{x}, \mathbf{y}\rangle = \int_D \mathbf{x}(t)\cdot\mathbf{y}(t)\mathrm{d}t.$$

Thanks to the relative abstractness of an inner product space, we will not have to make too many changes in moving into multivariate data.

Example 7. Roughness Penalties

Suppose we have a penalty of the form $\|\mathbf{x}''\|$. If we expand out the inner product we see $\|\mathbf{x}''\|^2 = \sum \int_D \mathbf{x}_k(t)^2 \mathrm{d}t$. Since we assume each component of $\mathbf{x}$ has its own basis function expansion, we can use the previous results on roughness penalties to find

$$\|\mathbf{x}\|^2 = \sum \mathbf{c}_i' \mathbf{K} \mathbf{c}_i.$$

Here as usual, $\mathbf{K}_{ij} = \langle \phi_i, \phi_j \rangle$ where we use the one-dimensional inner product $\langle f, g \rangle = \int_D fg \mathrm{d}t$.

Notice that our penalty decomposes into a sum of simpler penalties. This suggests we could take a multiple penalty approach. Instead of an expression of the form $\lambda\|\mathbf{x}\|^2$ we have a sum of penalties

$$\sum \lambda_k \|\mathbf{x}_k\|^2 = \lambda_1 \mathbf{c}_1' \mathbf{K} \mathbf{c}_1 + \cdots + \lambda_m \mathbf{c}_m' \mathbf{K} \mathbf{c}_m$$

Example 8. Generalised Roughness Penalties

In all the previous cases the weights λ have been *external* to the norms we used. What if we instead used a weighted inner product of the form $\langle x, y \rangle_Q = \mathbf{x}'\mathbf{Q}\mathbf{y}$? To define an inner product we must have that $\mathbf{Q}$ be symmetric and positive definite. However since we only have $\lambda \geq 0$ in general, we will only say that $\mathbf{Q}$ must be positive semidefinite and symmetric. We define a symmetric, positive semidefinite bilinear form, but still retain the inner product notation, $\langle \mathbf{x}, \mathbf{y} \rangle = \int_D \mathbf{x}'\mathbf{Q}\mathbf{y} \mathrm{d}t$.

What will the roughness penalty look like with this change? If $\mathbf{Q}$ is diagonal then we will get the results above with the multiple λ's.

In the case of a general suitable matrix, by making use of the usual approach we find

$$\begin{aligned} \|\mathbf{x}''\|_Q^2 &= \sum\sum q_{ij} \langle \mathbf{x}_i'', \mathbf{x}_j'' \rangle_{L^2} \\ &= \sum\sum q_{ij} \mathbf{c}_i' \mathbf{K} \mathbf{c}_j. \end{aligned}$$

Example 9. Lotka Volterra Equations

The Lotka Volterra Equations are a model of the interactions of a prey and predator population. They are as follows

$$\begin{aligned} x' &= x(\alpha - \beta y) \\ &= \alpha x - \beta xy \\ y' &= -y(\gamma - \delta x) \\ &= -\gamma y - \delta xy. \end{aligned}$$

We will be using the standard dot product for our norms. In the standard inner product, the different terms are independent of each other; we only need to look at one equation, so without loss of generality we will find a formula for

$$\|x' - \alpha x - \beta xy\|^2.$$

Here we are computing a "one dimensional" penalty. Notice we no longer assume anything about the signs of the coefficients. As usual the penalty is then expanded out as an inner product

$$\begin{aligned} \|x' - \alpha x - \beta xy\|^2 &= \langle x' - \alpha x - \beta xy, x' - \alpha x - \beta xy \rangle \\ &= \|x'\|^2 + \alpha^2\|x\|^2 + \beta^2\|xy\|^2 + 2\alpha\beta\langle xy, x \rangle - 2\alpha\langle x', x \rangle - 2\beta\langle x', xy \rangle. \end{aligned}$$

The "linear" terms such as $\langle x', \alpha x \rangle$ have already been covered. We will look at the nonlinear terms instead.

Firstly $xy = \sum\sum b_i\phi_i c_j \phi_j$. If we define $d = (b_1c_1, \ldots, b_nc_n)$ we get

$$\|xy\|^2 = \mathbf{d}'\mathbf{K}\mathbf{d}.$$

Where $\mathbf{K}_{ikjl} = \langle \phi_i\phi_k, \phi_j\phi_l\rangle$. In a similar manner we can write $\langle x', xy\rangle = \mathbf{b}'\mathbf{P}\mathbf{d}$, here $\mathbf{P}$ is only a $m \times m^2$ matrix because the space spanned by the products $\phi_i\phi_j$ is bigger than that spanned by the derivatives ϕ_i'. Its entries are given by the result $\mathbf{P}_{ijk} = \langle \phi_i, \phi_j\phi_k\rangle$. The exact same technique can be used to find $\langle x, xy\rangle$. Combining all these results

$$\|x' - \alpha x - \beta xy\|^2 = \mathbf{b}'\mathbf{K}\mathbf{b} + \alpha^2\mathbf{b}'\mathbf{L}\mathbf{b} + \beta^2\mathbf{d}'\mathbf{M}'\mathbf{d} + 2\alpha\beta\mathbf{d}'\mathbf{N}\mathbf{b} - 2\alpha\mathbf{b}'\mathbf{O}\mathbf{b} - 2\beta\mathbf{b}'\mathbf{P}\mathbf{d}.$$

8 Burger's Equation

The theory of nonlinear partial differential equations is not much harder than that of nonlinear ordinary differential equations. We will look at the dimensionless Burger's eqation

$$u_t + uu_x = \epsilon u_{xx}$$

This equation arises in the theory of shock waves. First we will derive the penalty for the inviscid Burger's Equation or the Riemann Equation

$$u_t + uu_x = 0$$

As was the case for multivariate splines, we expand u as a sum of basis functions $u(x,t) = c_1\phi_1(x,t) + \cdots + c_M\phi_M(x,t)$. This equation is simpler than the Lotka Volterra system. Define our inner product $\langle u, v\rangle = \int_D u(x,t)v(x,t)dxdt$. We must define $\|u_{tt} + uu_{xx}\|^2$. If as before we let $\mathbf{d} = (c_1c_1, \ldots, c_Mc_M)$ we can expand out the inner product.

$$\begin{aligned}\langle u_t + uu_x, u_t + uu_x\rangle &= \|u_t\|^2 + 2\langle u_t, uu_x\rangle + \|uu_x\|^2 \\ &= \mathbf{c}'\mathbf{K}\mathbf{c} + 2\mathbf{c}'\mathbf{L}\mathbf{d} + \mathbf{d}'\mathbf{M}\mathbf{d}\end{aligned}$$

As usual $\mathbf{K}_{ij} = \langle \partial_t\phi_i, \partial_t\phi_j\rangle, \mathbf{L}_{ijk} = \langle \partial_t\phi_i, \phi_j\partial_x\phi_k\rangle$ and $\mathbf{M}_{ijkl} = \langle \phi_i\partial_x\phi_j, \phi_k\partial_x\phi_l\rangle$.

Returning to the full Burger's equation we can see break up the penalty

$$\|u_t + uu_x - \epsilon u_{xx}\|^2 = \|u_t + uu_x\|^2 - 2\epsilon\langle u_t + uu_x, u_{xx}\rangle + \epsilon^2\|u_{xx}\|^2$$

We just derived an expression for the first term, and we have already covered the calculation of expressions of the form $\|u_{xx}\|^2$, so we only need to concentrate on the inner product term. Using the bilinearity of the inner product we need only find expressions for $\langle u_t, u_{xx}\rangle$ and $\langle uu_x, u_{xx}\rangle$.. The first term can be written as $\mathbf{c}'\mathbf{O}\mathbf{c}$, where $\mathbf{O}_{ij}\langle \partial_t\phi_i, \partial_x^2\phi_j\rangle$. The second term is of the form $\mathbf{c}'\mathbf{P}\mathbf{d}$, where $\mathbf{P}_{ijk} = \langle \partial_x^2\phi_i, \partial_x\phi_j\phi_k\rangle$.

9 Fisher's Equation

In one dimension Fisher's Equation takes the form

$$u_t = u(1-u) + u_{xx}$$

As was the case for Burger's Equation, we will simplify by dealing with the penalty piecemeal. If we remove the u_{xx} we get the logistic equation $u_t = u(1-u)$. By the usual methods we get

$$\begin{aligned}\|u_t - u(1-u)\|^2 &= \|u_t\|^2 + \|u\|^2 + \|u^2\|^2 + 2\langle u_t, u\rangle + 2\langle u_t, u^2\rangle + 2\langle u, u^2\rangle \\ &= \mathbf{c}'\mathbf{K}\mathbf{c} + \mathbf{c}'\mathbf{L}\mathbf{c} + \mathbf{d}'\mathbf{M}\mathbf{d} + 2\mathbf{c}'\mathbf{N}\mathbf{d} + 2\mathbf{c}'\mathbf{O}\mathbf{d} + 2\mathbf{d}'\mathbf{P}\mathbf{d}\end{aligned}$$

As before we can add in the u_{xx} term and we find that $\|u_t - u(1-u) - u_{xx}\|^2 = \|u_t - u(1-u)\|^2 - 2\langle u_t - u - u^2, u_{xx}\rangle + \|u_{xx}\|^2$ We have already encountered most of the terms already. The term $\langle u^2, u_{xx}\rangle$ is similar to the term $\langle uu_x, u_{xx}$ from Burger's equation and can be dealt with in a similar manner.

If the equation has constants, such that $u_t = ru(1-u) + Du_{xx}$ then the terms can be scaled, which we will demonstrate below.

This system of equations only applies to the line though. If we wish to genearlise to any euclidean space we have

$$u_{tt} = ru(1-u) + D\Delta u$$

The inner product must be extended to multiple dimensions: $\langle u, v\rangle = \int_D u(\mathbf{x}, t)v(\mathbf{x}, t)\mathbf{dx}\mathrm{dt}$. The penalty becomes

$$\|u_{tt} - ru(1-u) - D\Delta u\|^2 = \|u_{tt}\|^2 + r^2\|u\|^2 + r^2\|u^2\|^2 D^2\|\Delta u\|^2 + 2r\langle u_{tt}, u\rangle + 2r\langle u_{tt}, u^2\rangle + \\ 2D\langle u, \Delta u\rangle + 2r^2\langle u, u^2\rangle + 2rD\langle u, \Delta u\rangle + 2rD\langle u^2, \Delta u\rangle$$

We have already encountered most of these terms before. The prescence of the laplacian in place of the ∂_x^2 operator does not make things much more complicated from the point of view of manipulating these expressions. For example the terms $\langle u^2, u_{xx}\rangle$ and $\langle u^2, \Delta u\rangle$ are both in the form $\mathbf{c}'\mathbf{K}\mathbf{d}$, where $\mathbf{d} = (c_1c_1, \ldots, c_Mc_M)$ as usual. In the first case $\mathbf{K}_{ijk} = \langle \phi_i\phi_j, \partial_x^2\phi_k\rangle$ whilst in the latter $\mathbf{K}_{ij} - \langle \phi_i\phi_j, \Delta\phi_k\rangle$.

10 Tidal Bore Equations

The tidal bore equations are the following system of partial differential equations

$$u_t + uu_x + h_x = 0$$
$$h_t + hu_x + h_x u = 0$$

We write $u = \sum u_i\phi_i(x, t)$ and $h = \sum h_i\phi_i(x, t)$. We must calculate two penalties. We will cover the second one first. We see

$$\|h_t + hu_x + h_x u\|^2 = \|h_t\|^2 + \|hu_x\|^2 + \|h_x u\|^2 + 2\langle h_t, hu_x\rangle + 2\langle h_t, h_x u\rangle + 2\langle h_x u, hu_x\rangle$$

If we let $\mathbf{c} = (u_1, u_1, \ldots, u_M u_M), \mathbf{d} = (u_1h_1, \ldots, u_M h_M)$ and $\mathbf{f} = (h_1h_1, h_Mh_M)$, we can begin representing terms. The penalty can be written as

$$\mathbf{h}'\mathbf{K}\mathbf{h} + \mathbf{d}'\mathbf{L}\mathbf{d} + \mathbf{d}'\mathbf{L}\mathbf{d} + 2\mathbf{h}'\mathbf{M}\mathbf{d} + 2\mathbf{h}'\mathbf{M}\mathbf{d} + 2\mathbf{d}'\mathbf{N}\mathbf{d}.$$

In this case $\mathbf{K}_{ij} = \langle \phi_i, \phi_j\rangle, \mathbf{L}_{ijkl} = \langle \phi_i\partial_x\phi_j, \phi_k\partial_x\phi_l\rangle, \mathbf{N}_{ijk} = \langle \partial_t\phi_i, \phi_j\partial_x\phi_k$. Many matrices appear twice because of the fact we are using the same basis for u and h. This means that for example $h_x u$ and hu_x both look like linear combinations of terms such as $\phi_i\partial_x\phi_j$.

This phenomena also means that we can recycle these matrices for the first equation. The penalty is

$$\|u_t + uu_x + h_x\|^2 = \mathbf{u}'\mathbf{K}'\mathbf{u} + \mathbf{c}'\mathbf{L}\mathbf{c}' + \mathbf{h}'\mathbf{O}\mathbf{h} + 2\mathbf{h}'\mathbf{P}\mathbf{c} + 2\mathbf{h}'\mathbf{Q}\mathbf{u} + 2\mathbf{u}'\mathbf{R}\mathbf{c}.$$

The additional matrices represent inner products not present in the original system $\mathbf{O}_{ij} = \langle \partial_x\phi_i, \partial x\phi_j\rangle, \mathbf{P}_{ijk} = \langle \partial_x\phi_i, \phi_j\partial_x\phi_k\rangle, \mathbf{Q}_{ij} = \langle \partial_x\phi_i, \partial_t\phi_j\rangle, \mathbf{R}_{ijk} = \langle \partial_t\phi_i, \phi_j\partial_x\phi_k\rangle$.

11 Abstract Theory

Some of the definitions here could be given in a more general form, but they will be given here with the aim of proving specific results instead of building a whole theory.

Definition. The sum of two vector spaces V and W, $V + W$ is defined to be the set of all sums of them, $V + W = \{v + w | v \in V, w \in W\}$. The sum of n vector spaces $V_1 + \cdots + V_n$ is defined recursively as $(V_1 + \cdots + V_{n-1}) + V_{n-1}$.

If V and W share the same identity element, then V and W are subspaces of $V + W$. This can be seen since $V = \{v + 0 | v \in V\}$. This condition will always hold if V and W are subspaces of some other subspace.

Definition. The product $V_1 \times \cdots \times V_n$ of n vector spaces $\{V_1, \dots, V_n\}$ is defined as the set of n-tuples $(v_1, \dots, v_n), v_k \in V_k$.

Define an addition operation by $(v_1, \dots, v_n) + (w_1, \dots, w_n) = (v_1 + w_1, \dots, v_n + w_n)$, scalar multiplication by $\alpha(v_1, \dots, v_n) = (\alpha v_1, \dots, \alpha v_n)$ and an additive identity $0_{V_1 \times \cdots \times V_n} = (0_{V_1}, \dots, 0_{V_n})$ and we can form a new vector space.

We will be working in finite dimensional spaces, so any vector space V will have a basis $\{w_1, \dots, w_n\}$ such that any $v \in V$ can be written as $c_1 w_1 + \cdots + c_n w_n$. The scalars c_k are called coefficients or coordinates.

Theorem 1. *Let X be an m dimensional vector space, and Y be an n dimensional inner product space, with an inner product denoted by $\langle \cdot, \cdot \rangle$. Let $f : X \to Y$ be a function from X to Y. Let x be an arbitrary element of X and $\mathbf{x} = (x_1, \dots, x_m)$ be the vector of its coordinates. Then there exists a vector field $\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^n$ and an $n \times n$ matrix $\mathbf{K}$ such that $\|f(x)\|^2 = \langle f(x), f(x) \rangle = \mathbf{F}(\mathbf{x})'\mathbf{K}\mathbf{F}(\mathbf{x})$*

Proof. Every $y \in Y$ is determined by (or isomorphic to) its coordinates $(y_1, \dots, y_n)$, $f(x) \in Y$ is determined by x so we can write the coefficients of $f(x)$ as $(f_1(x), \dots, f_n(x) = (\mathbf{F}_1(\mathbf{x}), \dots, \mathbf{F}_n(\mathbf{x})) = \mathbf{F}(\mathbf{x})$. For any $v, w \in X$ we can expand out the inner product. Let $\{y_1, \dots, y_n\}$ be a basis for Y, then

$$\begin{aligned}
\langle v, w \rangle &= \langle \sum_{i=1}^{n} v_i y_i, \sum_{i=1}^{n} w_i y_i \rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle v_i y_i, w_j y_j \rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j \langle y_i, y_j \rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j k_{ij} \\
&= \mathbf{v}'\mathbf{K}\mathbf{w}
\end{aligned}$$

Here $\mathbf{v} = (v_1, \dots, v_n)$, $\mathbf{w} = (w_1, \dots, w_n)$ and $\mathbf{K}$ is a matrix whose entries are give by $\mathbf{K}_{ij} = \langle y_i, y_j \rangle$. Hence $\langle f(x), f(x) \rangle = \mathbf{F}(\mathbf{x})'\mathbf{K}\mathbf{F}(\mathbf{x})$.

□

In the context of functional data analysis, all our vector spaces are finite combinations of functions $c_1\phi_1 + \cdots + c_k\phi_k$. These spaces are all subspaces of $L^2(D)$ for some appropriate domain D and so the finite spaces are also inner product spaces; they inherit the inner product from $L^2(D)$. If we have an operator T acting on $L^2(D)$ and it maps linear combinations of functions onto linear combinations of functions, then we can appeal to the theorem and use it to find an expression for $\|Tf\|^2$, for $f = c_1\phi_1 + \cdots + c_k\phi_k$.

Example 10. If $Tf = f^2$, then $T(c_1\phi 1 + \cdots + c_2\phi_2) = c_1^2\phi_1^2 + c_1c_2\phi_1\phi_2 + \cdots + c_k^2\phi_k^2$. This operator maps the set spanned by the basis functions $\{\phi_k\}$ into the set spanned by their products $\{\phi_k\phi_l\}$, so we can use the above theorem to derive an expression for $\|f^2\|^2 = \int_D f^4\mathbf{dx}$. If we let $\mathbf{d} = (c_1c_1, c_1c_2, \ldots, c_kc_k)$, then it is given by $\mathbf{d}'\mathbf{K}\mathbf{d}$, where $\mathbf{K}$ is an $k^2 \times k^2$ matrix with $\mathbf{K}_{ijkl} = \langle\phi_i\phi_j, \phi_k\phi_l\rangle$.

If we have an operator T has this property, then it is trivial to see that αT also has this property as vector spaces are close under scalar multiplication. If there are two operators T and S that map a finite dimensional vector space V to subsets of the vector spaces $T(V)$ and $S(V)$ respectively. Then the sum of the two operators $T + S$ maps V to some subset of $T(V) + S(V)$. When we introduced the notation αT and $T + S$ we implicitly assumed that that our operators always come from some vector space, taking this into account we get a theorem.

Theorem 2. *If we have a vector space of operators on some set of functions, then the set of operators that map finite dimensional vector spaces to subsets of finite dimensional vector spaces form a vector subspace.*

Theorem 3. *If two operators T and S map every finite dimensional vector subspaces to subset of some other finite dimensional vector spaces, then so does their composition $T \circ S$, defined by $(T \circ S)f = T(Sf)$.*

Proof. By assumption, S maps a finite dimensional vector space V to a subset of a finite dimensional vector space $S(V)$, which is in turn mapped to a subset of a finite dimensional vector space by T. $\square$

Corollary 1. *If n operators $\{T_i\}$ map finite dimensional vectors spaces to subsets of finite dimensional vector spaces, so does their composition $T_1 \circ \cdots \circ T_n$*

Proof. We will work by induction. We have already shown the proposition to hold for $n = 2$. Assume that the proposition holds for $n = k - 1$. Then for $n = k$ write $T_1 \circ \cdots \circ T_k = (T_1 \circ \cdots \circ T_{k-1}) \circ T_k$. We know the proposition holds for T_k by assumption and for $T_1 \circ \cdots \circ T_{k-1}$ by the induction hypothesis, it follows from the $n = 2$ case that their composition also satisfies the proposition. $\square$

Definition. An n-linear operator is a linear operator on the product of n vector spaces $V_1 \times \cdots \times V_n$. A 2-linear operator is also called a bilinear operator.

Example 11. If we had two functions x and y in $L^2(D)$, then the m xy^2 would be 3-linear.