Engineering Computation Project
Modelling Wireless Communication Channels

Wojciech Dziwulski, Mansfield College

1 Introduction

This report explores the properties of various orthogonal function sets and their applicability to generating fits to real-life data.

2 Orthogonality

Orthogonality is a crucial mathematical concept and extends from the well-known geometrical application (vectors) to the analysis of functions.

We can define a set of orthogonal functions $g_0, g_1, \ldots, g_n$ whose linear combination can be used to fit an arbitrary function f:

$$f(x) = a_0 g_0(x) + a_1 g_1(x) + a_2 g_2(x) + \ldots + a_n g_n \tag{2.1}$$

We then define an inner product operation, for which, conveniently:

$$\langle g_n, g_m \rangle = 0 \quad \text{if} \quad n \neq m \tag{2.2}$$

So that if we compute the inner product of both LHS and RHS with some arbitrary g_k, like:

$$\langle f, g_k \rangle = a_0 \langle g_0, g_k \rangle + a_1 \langle g_1, g_k \rangle + \ldots + \boldsymbol{a_k \langle g_k, g_k \rangle} + \ldots + a_n \langle g_n, g_k \rangle \tag{2.3}$$

which lets us calculate the a_k coefficient as:

$$a_k = \frac{\langle f, g_k \rangle}{\langle g_k, g_k \rangle} \tag{2.4}$$

This result is crucial, as it demonstrates that the coefficients are independent of the size of the orthogonal basis used for the investigation i.e. if more functions are decided to be added, old coefficients do not have to be recomputed.

Investigation of orthogonal function fitting can be started with analysing the Fourier series. The set of sines and cosines making up the basis turn out to be orthogonal under integral operation over one period. Even though useful, Fourier series is quite a basic method for function fitting and will not be analysed in detail in this report. We will instead delve into the investigation of two particularly relevant orthogonal basis functions - Gram-Schmidt and Laguerre.

3 Gram-Schmidt functions

3.1 Process

The Gram-Schmidt process stems from a very simple logic for generating orthogonal functions - choosing a linearly independent basis, and adjusting the consecutive functions appropriately, making sure that every subsequent function is orthogonal to the previous one.
Hence having the linearly independent basis $v_0, v_1, v_2, \ldots, v_n$ we can compute the orthogonal functions $g_0, g_1, g_2, \ldots, g_n$ using:

$$g_0(x) = v_0(x) \tag{3.1}$$
$$g_1(x) = v_1(x) - e_{10} g_0(x) \tag{3.2}$$
$$g_2(x) = v_2(x) - e_{20} g_0 - e_{21} g_1(x) \tag{3.3}$$
$$g_3(x) = v_3(x) - e_{30} g_0 - e_{31} g_1(x) - e_{32} g_2(x) \tag{3.4}$$

As noted before, using the orthogonality property we can now compute the e coefficients:

$$g_n(x) = v_n(x) - e_{n0} g_0 - \ldots - \boldsymbol{e_{nm} g_m(x)} - \ldots - e_{n(n-1)} g_{n-1}(x) \tag{3.5}$$

Taking the inner products:

$$\langle g_n, g_m\rangle = \langle v_n, g_m\rangle - e_{n0}\langle g_0, g_m\rangle \quad - \ldots - \boldsymbol{e_{nm}\langle g_m, g_m\rangle} \quad - \ldots - e_{n(n-1)}\langle g_{n-1}, g_m\rangle \tag{3.6}$$

$$0 = \langle v_n, g_m\rangle - 0 \quad - \ldots - \boldsymbol{e_{nm}\langle g_m, g_m\rangle} \quad - \ldots - 0 \tag{3.7}$$

$$\Rightarrow e_{nm} = \frac{\langle v_n, g_m\rangle}{\langle g_m, g_m\rangle} \tag{3.8}$$

We can then divide each function by its norm (square root of its inner product with itself) in order to achieve an orthonormal set.

3.2 Monomial basis

The independent basis chosen is a set of monomials $v_0 = 1, v_1 = x, v_2 = x^2, \ldots, v_n = x^n$ which will be used to produce an orthogonal basis under operation $\int_0^\infty g_n g_m e^{-x} dx$.
We compute the polynomial coefficients by iteratively multiplying the previous orders' coefficient vectors by the e-scaling factor:

Listing 1: gs_polynomials.m

```
coefficients = zeros(n+1);
g_g_product = zeros([1 n+1]);

% Calculating the Gram Schmidt coefficients based on the previous orders
for order = 0:n
  row = order+1;
  coefficients(row, n+1-order) = ones(row,n+1-order);
  for power = 1:order
    v_g_product = gs_inner_product(ones(row,:), coefficients(row-power
        ,:),x);
    e_coeff = v_g_product/g_g_product(row-power);
    coefficients(row, :) = coefficients(row, :) + (-1)*e_coeff*
        coefficients(row-power, :);
  end
  g_g_product(row) = gs_inner_product(coefficients(row,:), coefficients(
      row,:), x);
end
```

The leading coefficient of the appropriate orthogonal polynomial is retrieved from the matrix of coefficients of the independent basis ("ones" in listing 1). Interestingly, the Gram Schmidt functions are stored as their polynomial coefficients instead of the y-values calculated over the supplied x-range.
The x-range used for the computation of the Gram Schmidt e factors was chosen so that the marginal improvement over the evaluation accuracy did not exceed 1 % of the function value, and turned out to be around 70.

All the functions were tested in the gs_polynomials_testbench.m file.

4 Laguerre functions

Laguerre polynomials are another example of an orthogonal set, under the inner product defined as:

$$\langle L_n^{(\alpha)}, L_m^{(\alpha)}\rangle = \int_0^\infty x^\alpha e^{-x} L_n^{(\alpha)}(x), L_m^{(\alpha)}(x) dx = \begin{cases} \frac{\Gamma(n+\alpha+1)}{\Gamma(n+1)} & \text{for} \quad m = n \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

We are going to explore two methods of computing their coefficients and subsequently fit them to a real dataset.

4.1 Rodrigues formula

Using the Rodrigues formula, the n-th associated Laguerre polynomial is:

$$L_n^{(\alpha)}(x) = \frac{1}{n!} x^{-\alpha} e^x \frac{d^n}{dx^n}(x^{n+\alpha} e^{-x}), \qquad n \in \mathbb{N}, \alpha \in \mathbb{R} \tag{4.2}$$

noting that after differentiation the $x^{-\alpha}$ and e^x cancel out, all we have to do is calculate the coefficients of the differentiated product using the Leibniz rule:

$$(f \cdot g)^{(n)} = \sum_{k=1}^{n} \binom{n}{k} f^k g^{n-k} \tag{4.3}$$

This is accomplished by first recursively calculating the x^n coefficients and then calculating them by the rest of the appropriate product:

Listing 2: rodrigues_laguerre.m

```
for k = 2:n+1
  coefficients(k) = coefficients(k-1) * (n + alpha - (k-2));
end

factor = factorial(n);
for k = 1:n+1
  coefficients(k) = nchoosek(n,k-1)*coefficients(k)*((-1)^(n-k+1))/
     factor;
end
```

Noting, however, that the Rodrigues formula is computationally inefficient due to the repeated calculation of the derivative coefficients, we turn onto a different method: the recursive method.

4.2 Recursive method

This method takes the advantage of the successive polynomial generation using the formula:

$$nL_n^{(\alpha)}(x) = (2n + \alpha + 1 - x)L_{n-1}^{(\alpha)}(x) - (n + \alpha - 1)L_{n-2}^{(\alpha)}(x) \tag{4.4}$$

with:

$$L_0^{(\alpha)}(x) = 1 \tag{4.5}$$

$$L_1^{(\alpha)}(x) = -x + \alpha + 1 \tag{4.6}$$

This is achieved by the fairly simple:

Listing 3: recursive_laguerre.m

```
for order = 2:n
  row = order + 1;

  first_coef = (2*(order)+alpha-1)/(order);
  second_coef = -((order)+alpha-1)/(order);

  coefficients(row, :) = first_coef * coefficients(row-1, :) - circshift
     (coefficients(row-1, :), [0,1])/(row-1);
  coefficients(row, :) = coefficients(row, :) + second_coef *
     coefficients(row-2, :);
end
```

Note the use of the *circshift* function in order to simulate multiplication by the coefficient vector by the variable x.
For $\alpha = 0$ the Laguerre plots using the method above looked exactly as expected.

4.3 Gram-Schmidt Laguerre comparison

It is easy to note that under $alpha = 0$ the inner product operations for Laguerre and Gram-Schmidt polynomials look identical. This motivated further investigation of both of the orthogonal sets which yielded: