Engineering Computation Project
Modelling Wireless Communication Channels

Wojciech Dziwulski, Mansfield College

1 Introduction

Performance of Wi-Fi receivers is heavily dependent on scattering, hence building a sound mathematical model for signal propagation is needed for improving its quality. This report investigates the modelling of probability mass functions using orthogonal function sets such as Gram-Schmidt or Laguerre.

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 we can 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 important, 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 only one example of an orthogonal basis 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: Computing the Gram-Schmidt polynomial coefficients

```
1  coefficients = zeros(n+1);  % Initiate a matrix for the polynomial coefficients
2  g_g_product = zeros([1 n+1]); % Initiate a matrix for the self products
3
4  % Calculating the Gram Schmidt coefficients based on the previous orders
5  for order = 0:n
6    row = order+1; % Polynomial of n'th order is in (n+1)'th row
7    coefficients(row, n+1-order) = ones(row,n+1-order); % Leading coefficient
8    for power = 1:order % Remaining expansion coefficients
9      v_g_product = gs_inner_product(ones(row,:), coefficients(row-power,:),x);
10     e_coeff = v_g_product/g_g_product(row-power);
11     coefficients(row, :) = coefficients(row, :) + (-1)*e_coeff*coefficients(row-
           power, :); % Multiply the lower order polynomial by the e_coeff and add
12   end
13   g_g_product(row) = gs_inner_product(coefficients(row,:), coefficients(row,:), x
       ); % Compute the square of the normalizing constant
14 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. This makes the model more robust and clearer as patterns emerge from the coefficient values later on.

To calculate the inner product defined above, the values of the integrand were evaluated over a prescribed x-range while the integral itself was calculated using the MATLAB trapezoidal integration function *trapz*.The x-range was chosen by observing the volume of the integrand used for the inner product calculation. The product of the fourth order polynomial and fifth order monomial (used for the calculation of fifth order coefficients) turned out to drop dramatically at $x \approx 30$ and hit -2.678×10^{-7} at $x = 50$ (figure 3.1). Hence the range from 0 to 70 was used for all the calculations in order to maintain a reasonable accuracy margin. The orthonormality of the subsequently normalized functions has been later confirmed which proves that Gram Schmidt polynomials could be used as an appropriate function basis for fitting. All the functions were then 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.

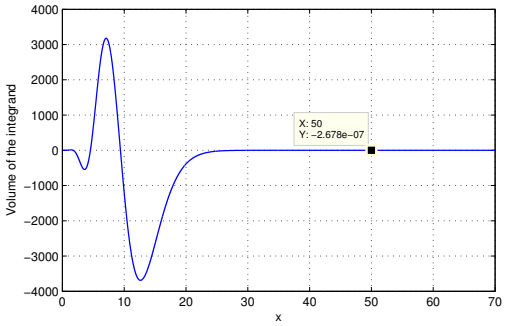


Figure 3.1: Product of the fourth order Gram Schmidt polynomial and fifth order monomial.

4.1 Rodrigues formula

Using the Rodrigues formula, the n-th associated Laguerre function 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: Computing the Laguerre polynomials using the Rodrigues formula

```
% Calculating the coefficient due to multiplied differentiation of an exponent
for k = 2:n+1
  coefficients(k) = coefficients(k-1) * (n + alpha - (k-2));
end

% Using the Leibniz rule for generalising the product rule in differentiation
factor = factorial(n); % Avoiding repeated factorial calculation
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: Computing the Laguerre polynomials using the recursive method

```
for order = 2:n
  row = order + 1; % Polynomial in n'th order is in (n+1)'th row
```

```
3
4       first_coef = (2*(order)+alpha-1)/(order); % Multiplier of the L_(n-1)
5       second_coef = -((order)+alpha-1)/(order); % Multiplier of the L_(n-2)
6
7       % Circshift simulates multiplying by 'x'
8       coefficients(row, :) = first_coef * coefficients(row-1, :) - circshift(
            coefficients(row-1, :), [0,1])/(row-1);
9       coefficients(row, :) = coefficients(row, :) + second_coef * coefficients(row-2,
            :);
10  end
11
12  coefficients = fliplr(coefficients); % Reflect the coefficients horizontally
```

Note the use of the *circshift* function in order to simulate multiplication of 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 an investigation of the similarities between the two orthogonal sets.

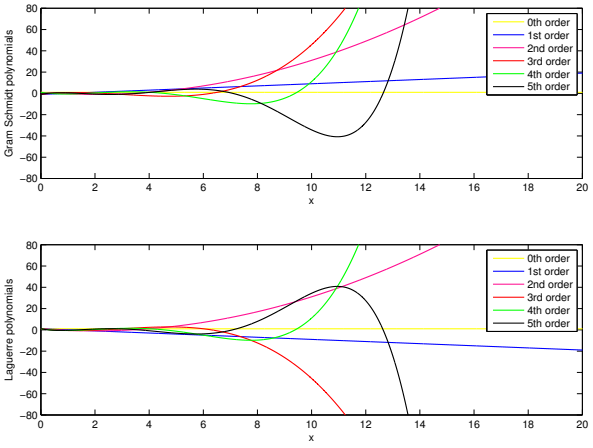


Figure 4.1: Comparison between the Gram Schmidt (red) and Laguerre (blue) polynomials up to order 5.

When plotted side by side (Figure 4.1) the Gram Schmidt and Laguerre polynomials indeed turn out to have much in common. It is clear that for some of the polynomial orders, the Laguerre plot is just the Gram Schmidt plot negated. This is even clearer when the polynomial coefficients for both are printed out:

$$gs_{coefficients} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0 & 1.0000 & -1.0000 \\ 0 & 0 & 0 & 0.5000 & -2.0000 & 1.0000 \\ 0 & 0 & 0.1667 & -1.5000 & 3.0000 & -1.0000 \\ 0 & 0.0417 & -0.6667 & 3.0000 & -4.0000 & 1.0000 \\ 0.0083 & -0.2083 & 1.6667 & -5.0000 & 5.0000 & -1.0000 \end{pmatrix} \tag{4.7}$$

$$laguerre_{coefficients} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0 & -1.0000 & 1.0000 \\ 0 & 0 & 0 & 0.5000 & -2.0000 & 1.0000 \\ 0 & 0 & -0.1667 & 1.5000 & -3.0000 & 1.0000 \\ 0 & 0.0417 & -0.6667 & 3.0000 & -4.0000 & 1.0000 \\ -0.0083 & 0.2083 & -1.6667 & 5.0000 & -5.0000 & 1.0000 \end{pmatrix} \tag{4.8}$$

The odd powers of Laguerre polynomials are just their Gram Schmidt counterparts negated. This may suggest that if the linearly independent basis on which the Gram Schmidt polynomials are built is changed, both orthogonal sets are going to be equal.

Given that finding, it seems like negating the coefficients of odd order monomials, i.e. constructing a linearly independent set $1, -x, x^2, -x^3, \ldots$ will produce Gram Schmidt coefficients identical to the Laguerre ones. This is also algebraically consistent with the formula for Gram Schmidt coefficients and yielded expected results when implemented in MATLAB.

4.4 Associated Laguerre functions

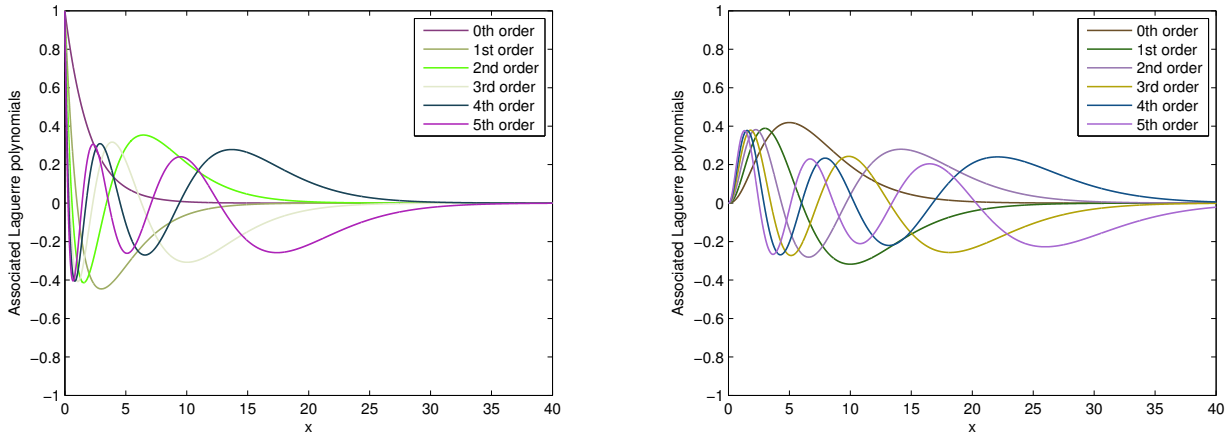
Finally, the set of **associated** Laguerre functions was produced using the formula:

$$\psi_n^{(\alpha)}(x) = \sqrt{\frac{\Gamma(n+1)}{\Gamma(n+\alpha+1)}} x^{\alpha/2} e^{-x/2} L_n^{(\alpha)}(x) dx \tag{4.9}$$

Associated Laguerre functions form an orthonormal set under the inner product

$$\langle f_n, f_m \rangle = \int_0^\infty f_n(x) f_m(x) dx \tag{4.10}$$

since after some algebraic manipulation it becomes analogous to the product in the equation 4.1 with the square root factor cancelled out.



(a) $\alpha = 0$ (b) $\alpha = 5$

Figure 4.2: Associated Laguerre functions to the 5th order

5 Synthesised Data fitting

5.1 Least squares Laguerre fit

We know that we can represent any function as a linear combination of the associated Laguerre functions. The coefficients of the functions in such combination can be easily calculated thanks to the orthogonality property under the inner product 4.10.
We wish to optimize our fit in the least square sense, though. Writing down our function as $f(x) = a_0\psi_0(x) + a_1\psi_1(x) + \ldots$, we can expand the general definition for the least squares error as:

$$e = \int_0^\infty (f(x) - f_0(x))^2 dx \tag{5.1}$$

$$= \int_0^\infty f^2 - 2\int_0^\infty f f_0 dx + \int_0^\infty f_0^2 \tag{5.2}$$

$$= \int_0^\infty (a_0)^2(\psi_0)^2 + a_0 a_1 \psi_0 \psi_1 + \ldots dx - 2\int_0^\infty (a_0\psi_0(x) + a_1\psi_1(x) + \ldots) f_0 dx + constant \tag{5.3}$$

$$= (a_0^2 + a_1^2 + \ldots) - 2\int_0^\infty (a_0\psi_0(x) + a_1\psi_1(x) + \ldots) f_0 dx + constant \tag{5.4}$$

And now optimizing for the coefficients:

$$\frac{\partial e}{\partial a_n} = 2a_n - 2\int_0^\infty \psi_n f_0 dx = 0 \tag{5.5}$$

so that indeed:

$$a_n = \int_0^\infty \psi_n f_0 dx \tag{5.6}$$

as suggested by the inner product definition quoted earlier (4.10), which proves that the Laguerre fit is such under the least squares sense.

5.2 Sample data

The sample fitting data set was generated using the supplied *exp_ data.m* function. The function generates a set of random complex numbers, scaled by the factor $\sqrt{\sigma^2/2}$ and shifted by the mean μ. Power of the dataset is then calculated by squaring the modulus of the generated values.
In order to plot the data, we generate a histogram with a prescribed number of bins which represent equally spaced intervals. The number of power data samples belonging to each interval is then calculated, with the whole dataset subsequently normalized to approximate a probability density function.

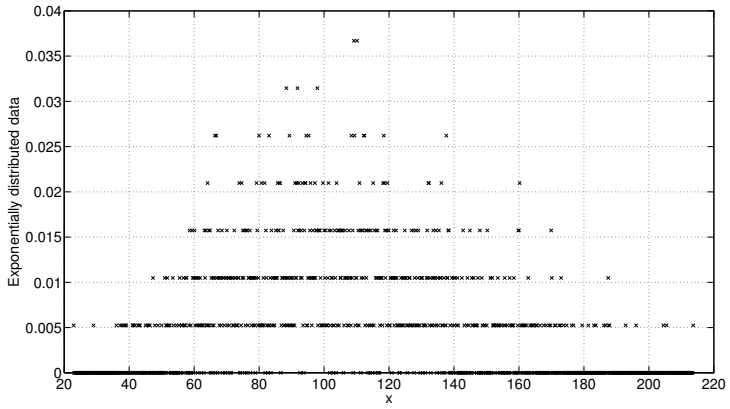


Figure 5.1: Exponentially distributed data.

The data shown in figure 5.1 was generated using $\sigma^2 = 5, \mu = 10, n_{samples} = 1000, n_{bins} = 1000$. A distinctive feature of the set is "banding" of the data points, stemming from the fact that the non-normalized number of samples in a bin is always an integer, often the same for different bins. To obtain a less scattered graph the number of bins could thus be reduced.

5.3 Basic Laguerre fit

Having a sound dataset generated, the applicability of Laguerre function fitting could now be tested. The function *laguerre_fit.m* thus takes the four inputs:

- fo - the function values
- x - the domain for which the function values were generated
- n - the highest desired order of the fitting Laguerre function
- alpha - the α coefficient

and returns the **fitted data points** as an output.

Listing 4: Fitting the dataset using the Laguerre basis

```
1 function result = laguerre_fit(fo, x, n, alpha)
2   lag_values = associated_laguerre(n, alpha, x); % Obtaining the fitting functions
3   coefs = zeros(1,n+1); % Multipliers of the appropriate functions
```

```
  for order = 0:n
    row = order+1; % Function of n'th order is in (n+1)'th row
    coefs(row) = laguerre_inner_product(lag_values(row,:), fo, x); % Fitting
       coefficients
  end
  result = coefs * lag_values; % Multiplying the function values by appropriate
     coefificients and adding up
end
```

The code listing 4 numerically evaluates the inner product between the supplied dataset and the Laguerre function and then multiplies it by the data points of the said function.

Figure 5.2 presents the generated data points for $\sigma^2 = 2, \mu = 0, n_{samples} = 10^4, n_{bins} = 100$. The plot looks like a fast decaying exponential, since a customary distribution "bell" is placed on the origin, hence points on the LHS of the axis are reflected and added to the RHS of the distribution. The best fit line (blue) clearly shows that even though the fit further from the origin can be considered reasonable, it cannot cope with a sharp exponential increase near x=0. This is because, as seen in figure 4.2, all of the 5th order Laguerre polynomials pass through the origin, hence cannot model a non-zero function value there.

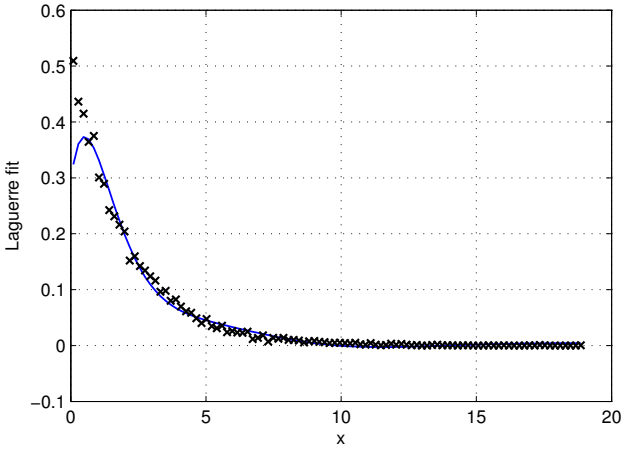


Figure 5.2: Basic 5th order, $\alpha = 0$ Laguerre fit for $\sigma^2 = 2, \mu = 0, n_{samples} = 10^4, n_{bins} = 100$.

5.4 Parameterised fit comparison

Realising the mathematical origin of the randomly scattered data gives us a way of testing the accuracy of the fit. Our generated data can be described by means of a random variable X which is a sum of squares of two other random variables. Both have an identical variance, which means that X is chi-squared distributed. The probability density function of the chi-squared distribution is:

$$f(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}, \quad x \in (0, \infty) \tag{5.7}$$

where, in our case, the number of degrees of freeedom n, equals 2, hence:

$$f(x) = \frac{1}{2} e^{-x/2} \tag{5.8}$$

Given our definition of the Laguerre functions (eq. 4.9), an $\alpha = 0$, 0th order fit should provide an exact match for the above pdf. To compare it to the expectation, we construct a parameterised fit of the form $f(x) = \frac{1}{\xi} e^{-x/\xi}$. The mean ξ is estimated by using the fact that for a probability distribution function $f(x)$ the mean is $\xi = \int_0^\infty x f(x) dx$.

The error, in the least squares sense, between the Laguerre and parameterised fit (figure 5.3), was evaluated to be 2.30×10^{-4}. Contrary to the expected 0.5, the calculated Laguerre leading coefficient is 0.4845 which justifies the discrepancy. The parameterised fit is plotted based on the estimated mean of $\xi = 2.0554$ - clearly, a lot closer to the desired mean of 2. The parameterised fit seems to match the data more closely in the intermediate values of x, but the error between the two functions is insignificant - three orders of magnitude smaller than the largest function values.

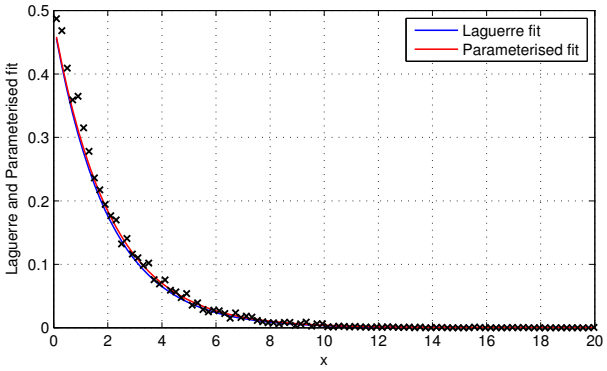
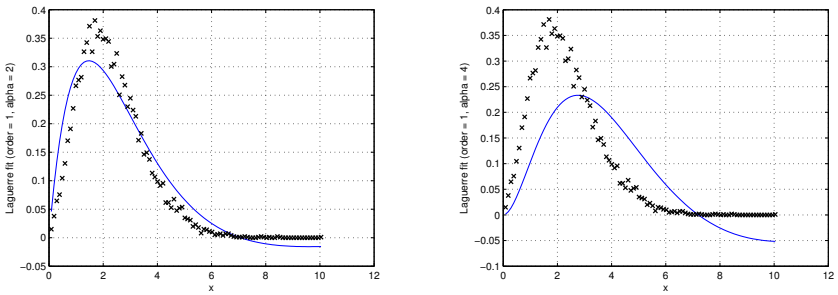


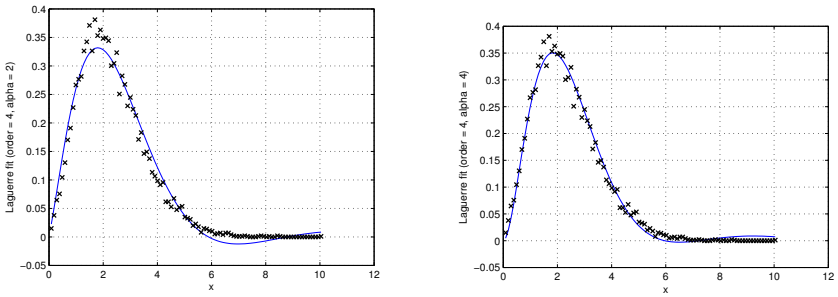
Figure 5.3: Exponentially generated data ($\sigma^2 = 2, \mu = 0, n_{samples} = 10^4, n_{bins} = 100$), ideal parameterised fit (red) and the 0-th order, $\alpha = 0$ Laguerre fit (blue).

5.5 Difficult data

It is clear that an orthonormal set of functions is best used for fitting in the range that it is defined. Fits for different orders and values of α are presented in figure 5.4.



(a) $n = 1, \alpha = 2$ (b) $n = 1, \alpha = 4$



(c) $n = 4, \alpha = 2$ (d) $n = 4, \alpha = 4$

Figure 5.4: Laguerre fits of different orders and alpha parameters

As we can see, the higher order fits also provide a closer matching of the generated data. It is unclear from the figures, though, how alpha influences this accuracy - in case of the 0th order function bigger alpha deteriorated the fit, whereas for the 4th order it improved the fit noticeably. A possible explanation for the inaccuracy is the integration range - for the Laguerre inner product 0 to ∞ and in practice the x range computed by the *exp_ data.m* function.

It is worth noting that if the data is shifted even further, the Laguerre functions provide an even worse fit. Figure 5.5 clearly shows that Laguerre fit is of nearly no use even for a mean of 5.

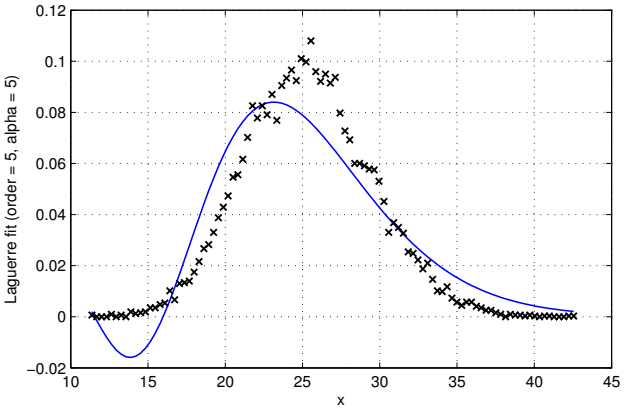


Figure 5.5: A 5th order Laguerre fit for $\alpha = 5$ with data generated for $\sigma^2 = 1/3, \mu = 5, n_{samples} = 10000, n_{bins} = 100$

5.6 Shifting and scaling

Fitting a set of functions to data is only possible if both span a similar domain. Referring back to figure 4.2, the Laguerre polynomial values are defined on a range from 0 to 40, nearing zero everywhere after. It is clear that even though the supplied dataset might be defined on a domain much larger than the desired Laguerre one, we can shift and scale it to match our needs.

Shifting is somewhat easy - since the Laguerre polynomials are best defined on the above interval, we should bring the dataset as close as possible to zero. This is done by subtracting the smallest x coordinate in the set from all the other values in the x range:

$$x'(n) = x(n) - x(1) \tag{5.9}$$

Scaling of the set is more subtle. To have a sound factor to optimize, we first scale the dataset to be defined on the range $(0, 1)$ like:

$$x'(n) = \frac{x(n) - x(1)}{x(n_{max}) - x(1)} \tag{5.10}$$

and then scale it further by a factor λ:

$$x'(n) = \lambda \times \frac{x(n) - x(1)}{x(n_{max}) - x(1)} \tag{5.11}$$

Since the degree of the fit is arbitrarily chosen, the choice of λ should be tailored to the particular domain of the value set as well as the Laguerre order. The other factor requiring optimisation is α, whose value has been arbitrarily picked before, which is however crucial for the overall accuracy of the fit.

Both were simultaneously optimised using the fminsearch MATLAB function. **fminsearch** follows the Nelder-Mead method for finding a minimum of a function in a multidimensional space. The Nelder-Mead method, also called the amoeba algorithm, constructs a simplex in an n-dimensional space and tries to find an extremum of the given space. The algorithm iteratively and greedily replaces the "worst" point of the simplex through simple moves. Those are restricted to the reflection (through the centroid), expansion, contraction and reduction of the simplex.

In our case, the cost function requiring optimization was the least-squares error between the generated data and the fit. The implementation in listing 5 does exactly that - it first shifts and scales the domain, then computes the Laguerre fit and calculates the error.

Listing 5: Computing the cost function to optimise the fit

```
1  alpha = params(1); % Extracting the alpha coefficient
2  lambda = params(2); % Extracting the lambda coefficient
3
4  % Computing the scaled domain. We shift the x values to the left as much
5  % as possible and scale it by a chosen factor
6  xscaled = lambda*(x-x(1))/(x(length(x))-x(1));
7  laguerre = laguerre_fit(fo, xscaled, n, alpha);
8
9  % Computing the error
```

```
10  error = trapz(x, (laguerre-fo).^2);
```

The error is then used in the laguerre_optimal_fit function which returns the values of the actual minimum-error fit. The fminsearch requires supplying the initial guesses for the parameter values, which were chosen to be 5 and 10 for alpha and lambda respectively.

The choice of the initial guesses was based on the empirical observations. The scaling parameter corresponds to the approximate middle of the interval for which the associated Laguerre functions are greater than 0. Comparing the Associated Laguerre functions for $\alpha = 0$ and $\alpha = 5$ in figure 4.2, it is clear that the greater the alpha, the more right-shifted the maxima of the functions appear to be. That is why for the initial scaling factor of 10, an initial guess of $\alpha = 5$ is appropriate. It should be noted, however, that for $\alpha > 0$ the functions' values at the origin are 0 (due to the $x^{\alpha/2}$ factor). Hence the datasets were then fitted with both $\alpha = 0$ and $\alpha = 5$ with the difference between the fit errors compared for the two outcomes and more optimal option chosen.

Listing 6: Optimising the Laguerre fit

```
1  % Initial estimates for the values of alpha and lambda
2  alpha_guess = 5;
3  lambda_guess = 10;
4
5  % Optimisation of alpha and scale using fminsearch
6  params = fminsearch(@(params) fitting_error(fo, x, n, params), [alpha_guess,
       lambda_guess]);
```

The method described above has been tested on numerous datasets generated using the *exp_data.m* function and proved to be extremely accurate - both for highly spread and narrow values regardless of their distance from the origin.

6 Real Data fitting

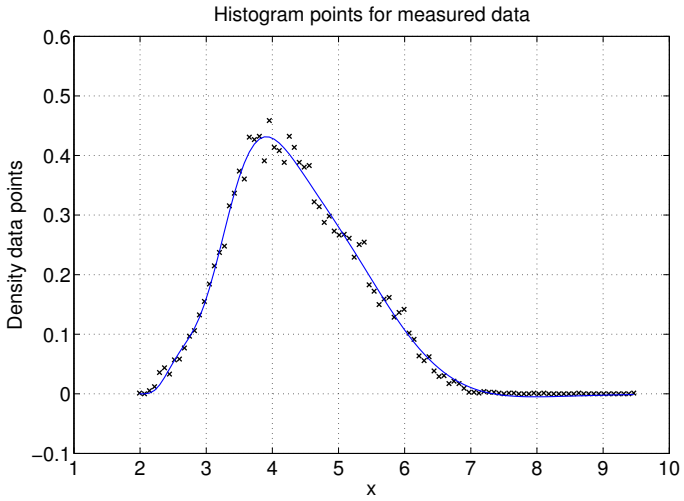


Figure 6.1: A Laguerre fit to the measured dataset

As seen in figure 6.1, the fitting algorithm described above yields an excellent match between the measured data and the best fit line. The error between the two is 1.3×10^{-3} for the 5th order fit.

7 Conclusions

The fitting procedure proved to be accurate for various datasets - close and far from the origin, both with high and low degree of scattering. The fitting proves to be extremely useful for statistical modelling of the obtained data. Computing the mode of the distribution corresponds to the position for which the power is maximum. The standard deviation will be useful for measuring the dispersion, or scattering, of the Wi-Fi signal. These can be used to design either more focused routers with a higher degree of penetration, or ones whose signal can spread through larger areas. Either way, more insight into spreading of the signal gives us a chance to design better, tailored communication systems.