Report on

Basis Functions

by

Niranjan Ashok Jahagirdar	2017B3A70454P
Pranay Khariwal	2017B3A70565P

as part of the course CS F320 Foundations of Data Science,

under the supervision of Dr. Navneet Goyal.

Introduction to Basis Functions

Suppose that we want a mathematical description or an approximation of a curve or any other data distributed over space, time, etc. Some data might not be linearly separable. In both regression and classification problems, we are trying to approximate a curve. Flexibility is a central issue since we usually cannot say in advance how complex the curve will be or specify certain of its characteristics. We don't have the time or patience to search some handbook of known functions for one that looks like what we want to study.

We need a set of basic functional building blocks that can be stacked on top of one another so as to have the features that we need. In other word we need *mathematical legos.* These mathematical legos are also called as **basis functions**. The symbol $ø_k(t)$ to stand for the kth function, and we call this the kth *basis function.*

By "stacking" in mathematics we mean adding things, possibly after multiplying each of them by its own constant. So here is how we will construct a function f(t) using K of these blocks:

$$f(t) = a_1ø_1(t) + a_2ø_2(t) + ... + a_kø_k(t)$$

The coefficients $a_1,a_2,\ldots,a_k$ determine the shape of the function.

What do we want from the basis functions? What properties should they have?

1. We should be able to **compute the individual basis functions in a fast manner**. We may be using many basis functions in some situations. Thus we need to be able to evaluate $ø_k(t)$ in a fast manner. This is why polynomials were so important in the days of hand calculation. Modern methods of computation also permit the rapid evaluation of sines, cosines, and other transcendental functions.

2. The basis functions **should be flexible**. Polynomials can track such features only by using a large number of basis functions, but splines are much better at fitting highly curvy data. Fourier series aren't great at capturing sharp changes, but wavelets are terrific at this.

3. We should be able to **compute the coefficients in a fast manner**. Coefficients are calculated by solving a system of linear equations, with one equation for each basis function. This isn't hard if there is under a hundred of these, but if thousands are required, only basis functions systems that imply special structures for these equations will serve. Again, splines are excellent, but polynomials, Fourier series and wavelets will only work for relatively specialized situations, such as equally spaced values of *t*.

4. The basis functions **should be differentiable**. Derivatives play many roles in data modelling, and we will want basis functions that have well-behaved derivatives up to whatever order that we require. Fourier series certainly work here, as do splines. But polynomials fall down in the sense that their derivatives exhibit progressively more simple behavior the higher the order, and the real world tends not to behave like this.

5. The basis functions **should be constrained** in terms of the given monotonicity, positivity, periodicity, asymptotes, etc. Some functions can only be positive, others may be required to be strictly increasing, and we may want functions to become more and more like straight lines for extreme values of *t*. These constraints also need to be built into our basis systems. A common example is *periodicity*, and we have already noted that Fourier series, as well as wavelets, are the basis systems of choice in this case.

Different types of basis functions

1. Polynomial basis functions

$$f(x) = \sum_{i=0}^{k} a_i \varphi_i(x) \quad where \quad \varphi_i(x) = x^i$$

Polynomials, though simple and familiar, are actually not that flexible, and consequently they have been more or less replaced by a basis system called the *spline* basis system.

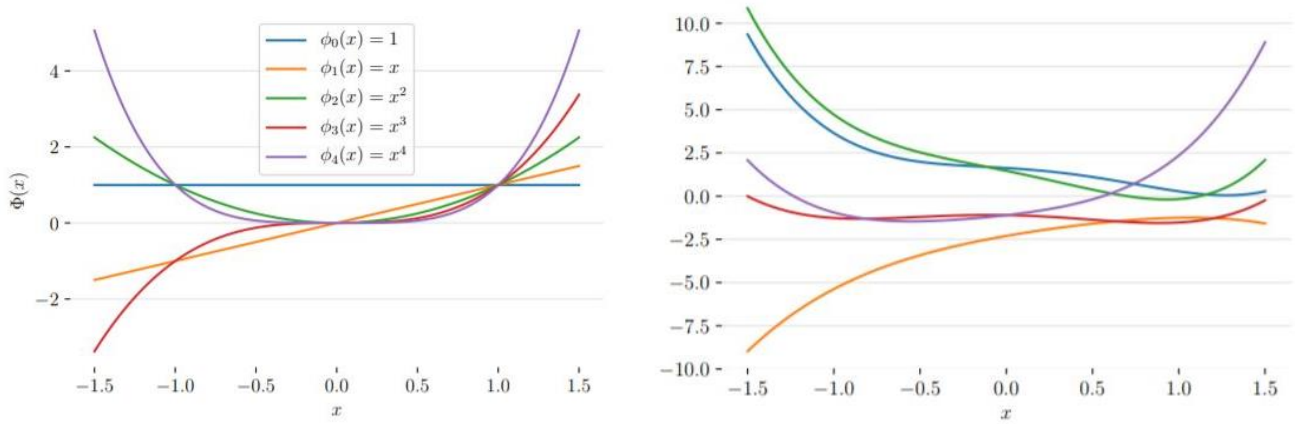


Fig 1. On the left we have polynomial basis functions upto power 4 and on the right, we have randomly assigned weights for the same functions.

C++ code implementation for polynomial functions

2. Fourier basis functions

$$f(x) = \sum_{i=0}^{k} a_i \varphi_i(x)$$

$$\text{where} \quad \varphi_0(x) = 1 \quad \text{and for j} > 0 \quad \varphi_i(x) = \cos(\omega_i x + \psi_i)$$

Joseph Fourier suggested that functions could be converted to a sum of sines and cosines. A Fourier basis is a linear weighted sum of these functions. Fourier basis functions **are good approximations for periodic** functions as these sine and cosine functions are periodic with period ω.

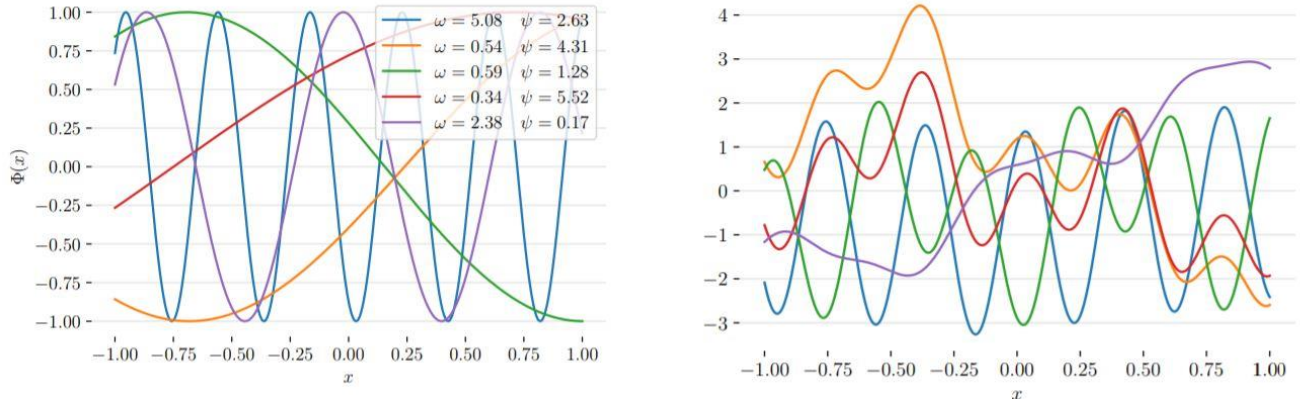


Fig 2. On the left we have cosine basis example with several arbitrary frequencies and phases and on the right, we have five random weightings of these functions.

C++ code implementation for Fourier functions

3. Gaussian basis functions

$$f(x) = \sum_{i=0}^{k} a_i \varphi_i(x)$$

where $\varphi_i(x) = e^{-0.5\left((x-\mu)/\sigma\right)^2}$ for a particular μ and σ

These are functions which are symmetric around a certain point and taper to zero on either side.

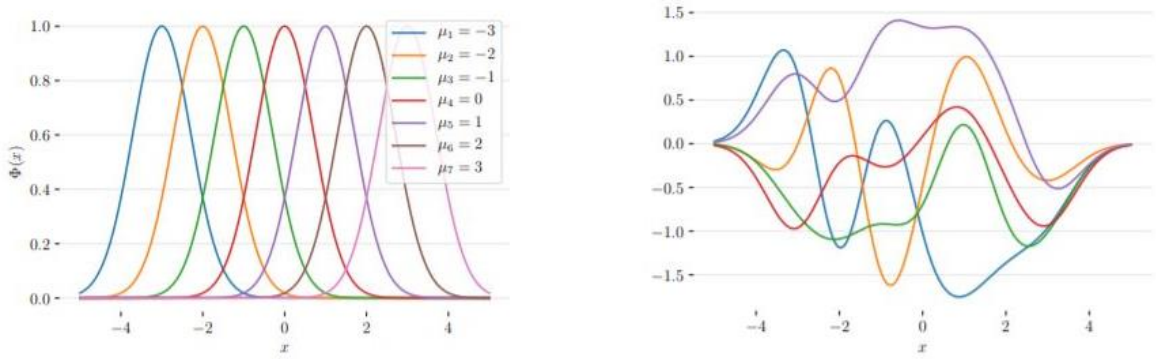


Fig 3. On the left we have Gaussian basis example with several different means and on the right, we have five random weightings of these functions.

C++ code implementation for Gaussian functions

4. Sigmoidal basis functions

$$f(x) = \sum_{i=0}^{k} a_i \varphi_i(x)$$

where $\varphi_i(x) = \dfrac{1}{1 + e^{(-c_i x)}}$

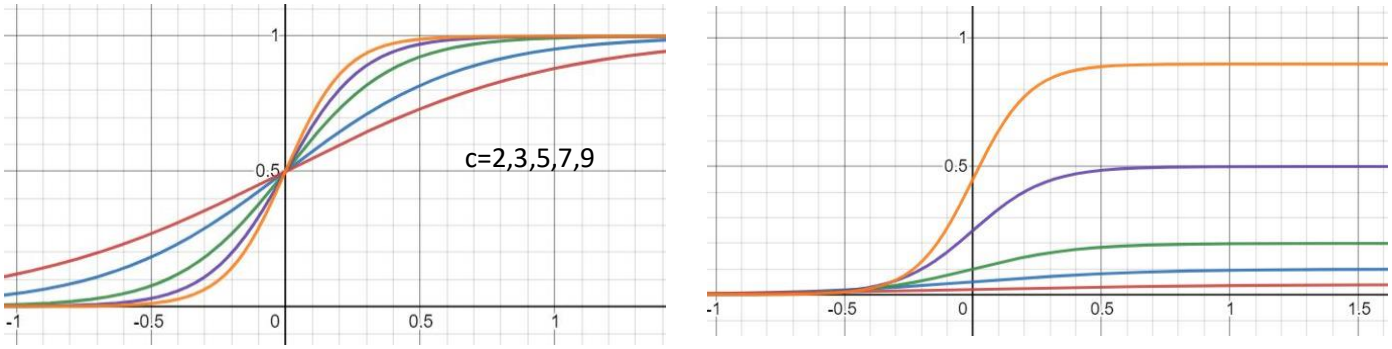


Fig 4. On the left we have sigmoidal functions with different c and on the right, we have five random weightings of these functions.

c is a positive constant which controls the slope or steepness of the sigmoid, and the sigmoid function amplifies and limits the small activation levels and high activation levels respectively.

C++ code implementation for sigmoidal functions

5. Spline and B-spline functions

Spline functions are piecewise polynomial functions in which the individual polynomials have the same degree and connect at joining points whose abscissa values, called **knots,** are pre specified.

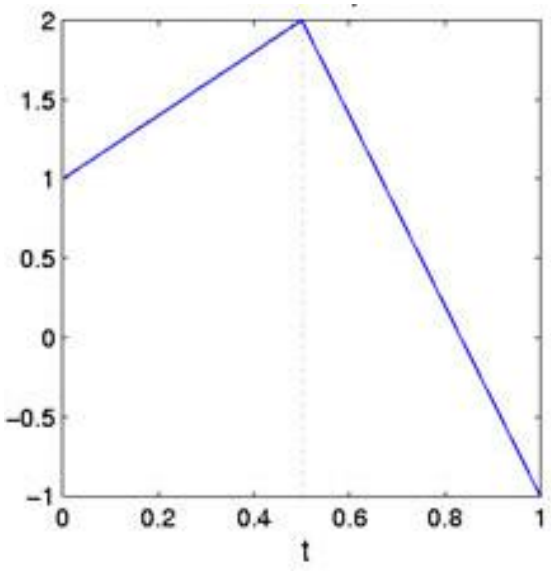


Fig 5. Spline of Order 2

In Fig. 5 we see a spline function of order 2 where the knot is at t=0.5. All the polynomial functions are of degree 1. Here we can see that at the knots only the values of the function f(t) are continuous while the first order derivatives of f(t) are not continuous.

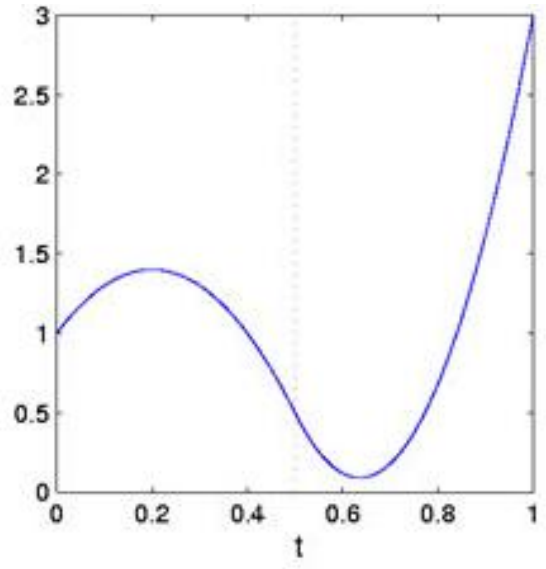


Fig 6. Spline of Order 3

In Fig. 6 we see a spline function of order 3 where the knot is at t=0.5. Here all the polynomial functions are of degree 2. We can see that at the knots, the values *and* the first order derivative of the function f(t) are continuous.

Similarly if we have a spline of order 4, the function, its first derivative as well as its second derivative will be continuous. Mathematically, for a spline of order 4 (i.e. polynomials will be of degree 3):

$$f(x+\in) = f(x-\in)$$

$$f'(x+\in) = f'(x-\in)$$

$$f''(x+\in) = f''(x-\in)$$

$$where\ x\ is\ a\ knot\ point\ and\ \in \rightarrow 0.$$

Let there be *n* knots $k_1 < k_2 <$…..<kn with degree d>=0 such that *s(x)* has d-1 continuous derivatives. Therefore, at each knot *d-1* order derivatives exist.

$$s(x) = P_o(x)\ for\ x < k_1,$$

$$= P_1(x)\ for\ k_1 \leq x \leq k_2,$$

$$= P_2(x)\ for\ k_2 \leq x \leq k_3,$$

.

.

$$= P_n(x)\ for\ x \geq k_n.$$

For a given sequence of knots, there is, up to a scaling facto, a unique spline $B_{i,n}(x)$ satisfying:

$$B_{i,n}(x) = 0 \quad if \quad x < k_i \quad or \quad x > k_{i+n}$$
$$= \ nonzero, otherwise$$

If we add the constraint that $\sum B_{i,n}(x) = 1$ for all x between the first and last knot, then the scaling factor of $B_{i,n}(x)$ becomes fixed. The resulting $B_{i,n}(x)$ spline functions are called B-splines.

Given a knot sequence …,t0,t1,t2,… the B-splines of order 1 are defined by:

$$B_{i,1}(x) = 1 \ \ if \ \ t_i \leq x < t_{i+1}$$
$$= 0 \ \ otherwise$$

The higher order B-splines are defined by recursion:

$$B_{i,k+1}(x) = w_{i,k}(x)B_{i,k}(x) + \left[1 - w_{i+1,k}(x)\right]B_{i+1,k}(x)$$
$$where \quad w_{i,k} = \frac{x - t_i}{t_{i+k} - t_i}, t_{i+k} \neq t_i$$
$$= \quad 0, \ \ otherwise$$

C++ code implementation for spline functions

6. Wavelets

Unlike a Fourier decomposition which always uses complex exponential (sine and cosine) basis functions, a wavelet decomposition uses a time-localized oscillatory function as the analyzing or mother wavelet. The mother wavelet is a function that is continuous in both time and frequency and serves as the source function from which scaled and translated basis functions are constructed. The mother wavelet can be complex or real, and it generally includes an adjustable parameter which controls the properties of the localized oscillation. Wavelet analysis is more complicated than Fourier analysis since one must fully specify the mother wavelet from which the basis functions will be constructed.

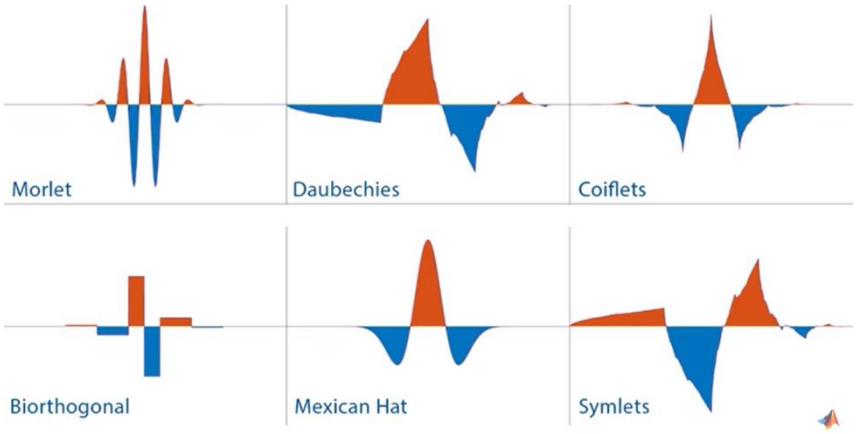


Fig 7. Few examples of wavelets

Let us look at morlets. In this:

$$f(x) = \sum_{i=0}^{k} a_i \varphi_i(x)$$

$$\text{where } \varphi_i(x) = \cos(\omega x + \alpha) e^{\left(\frac{-x^2}{2\sigma^2}\right)}$$

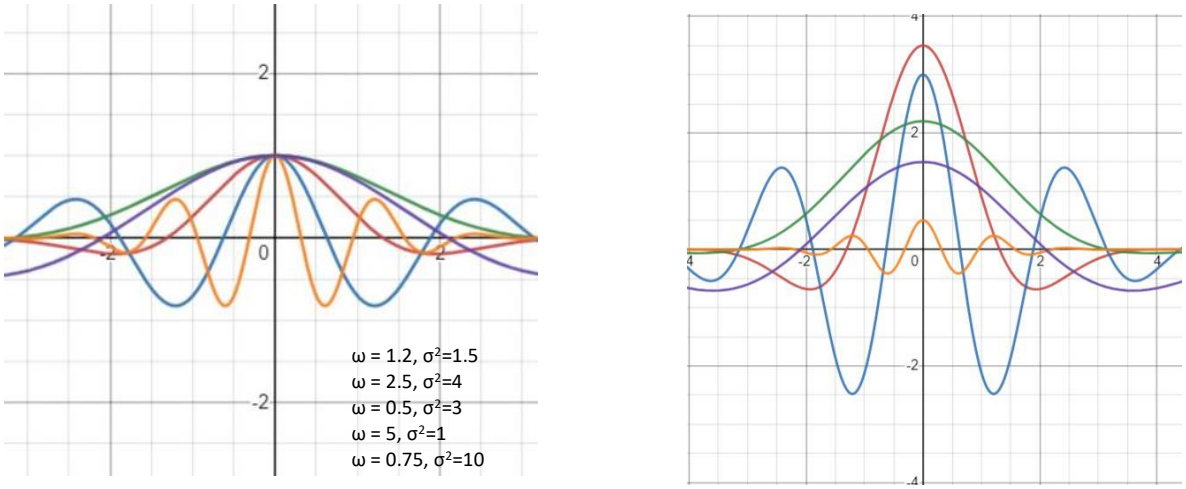


Fig 8. On the left we have morlet wavelets with different ω and σ while on the right, we have five random weightings of these functions.

C++ code implementation for morlets

How do we construct basis functions?

We start with a prototype function φ(t) known as the mother function.

We can apply three operations to it:

1. **Lateral shift φ(t+α)** to focus fitting power near a position α.
2. **Scale change φ(bt)** to increase resolving power for the capacity for local curvature.
3. **Smoothing** to increase differentiability and smoothness.

Running basis functions on sample data

Given below is the dataset that gives the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s and published in Ezekiel, M. (1930) Methods of Correlation Analysis, Wiley.

Speed (in mph) when the brakes were applied	Distance (in ft) it took the vehicle to stop
4	2
4	10

7	4
7	22
8	16
9	10
10	18
10	26
10	34
11	17
11	28
12	14
12	20
12	24
12	28
13	26
13	34
13	34
13	46
14	26
14	36
14	60
14	80
15	20
15	26
15	54
16	32
16	40
17	32
17	40
17	50
18	42
18	56
18	76
18	84
19	36
19	46
19	68
20	32
20	48
20	52
20	56
20	64
22	66
23	54
24	70
24	92
24	93

24	120
25	85

Plotting the data

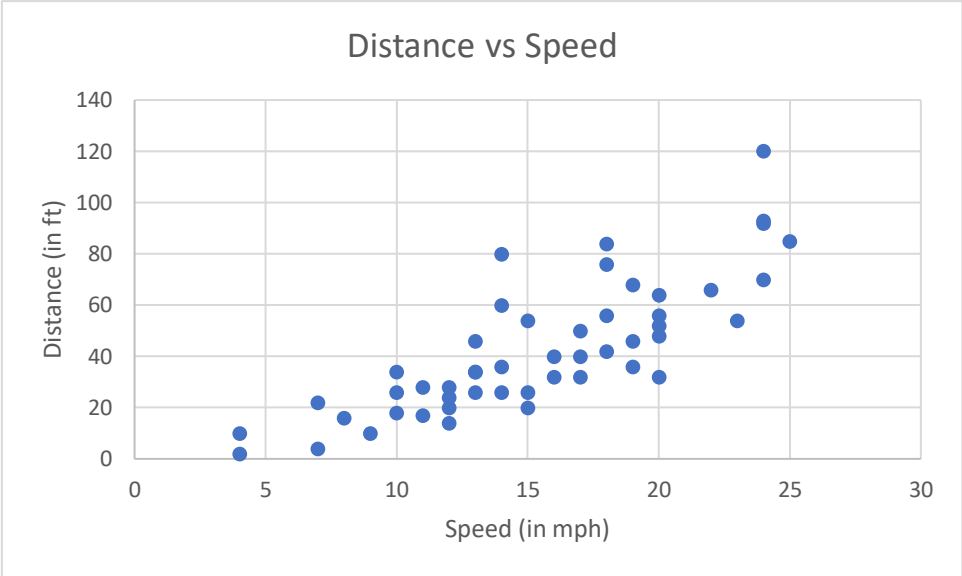


Fig 9. Data in the form of the scattered plot

Which basis function to use for the data shown in Fig 8.?

1. Data does not seem to be periodic. That rules out Fourier functions.

2. Data does not seem to be normally distributed in the sense that it is not symmetric or tapering. That rules out Gaussian functions.

3. Similarly from the shapes we can rule out sigmoidal functions.

4. **Polynomial functions** seem to be the best bet from the shape. In addition to this, we know from physics that distance, speed, acceleration etc. are often shown to be related in a linear way.

Let x = speed (in mph) when the brakes are applied. f(x) = distance (in ft) it took the vehicle to stop.

$$\mathrm{f}(x) = \sum_{i=0}^{k} a_i \varphi_i(x) \quad where \;\; \varphi_i(x) = x^i$$

Let k = 1. Degree of polynomial = 1.

$$f(x) = 3.9324x - 17.579$$

k = 2. Degree of polynomial = 2.

$$f(x) = 0.1x^2 + 0.9133x + 2.4701$$

k = 3. Degree of polynomial = 3.

$$f(x) = 0.0103x^3 - 0.3497x^2 + 6.8011x - 19.505$$

k = 4. Degree of polynomial = 4.

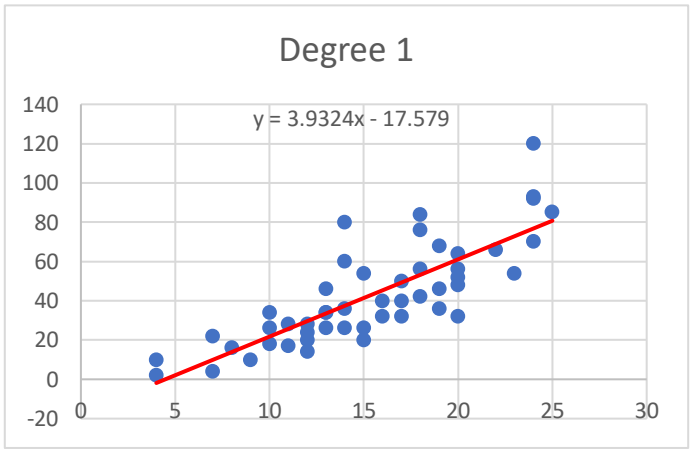
$$f(x) = 0.0028x^4 - 0.152x^3 + 2.8922x^2 - 18.962x + 45.845$$

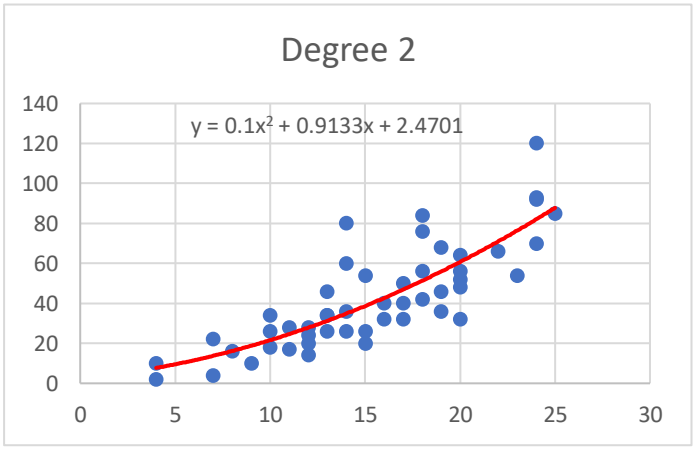
k = 5. Degree of polynomial = 5.

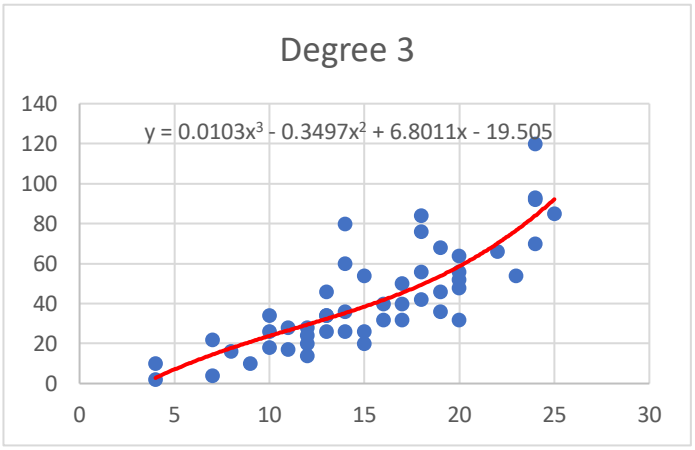
$$f(x) = 0.0002x^5 - 0.01x^4 + 0.194x^3 - 1.4261x^2 + 5.4843x - 2.6501$$

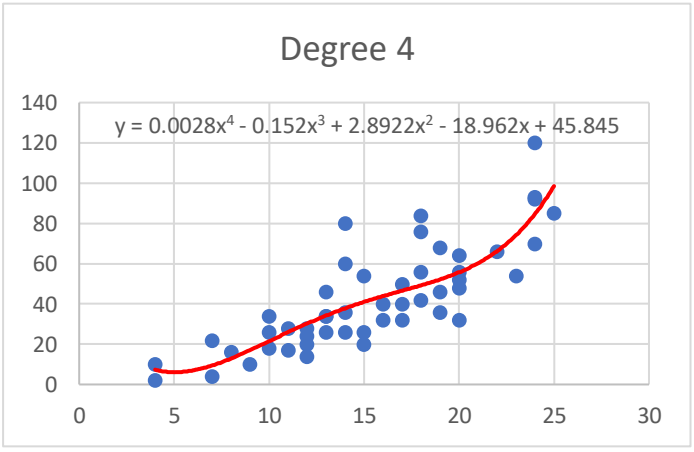
k = 6. Degree of polynomial = 6.

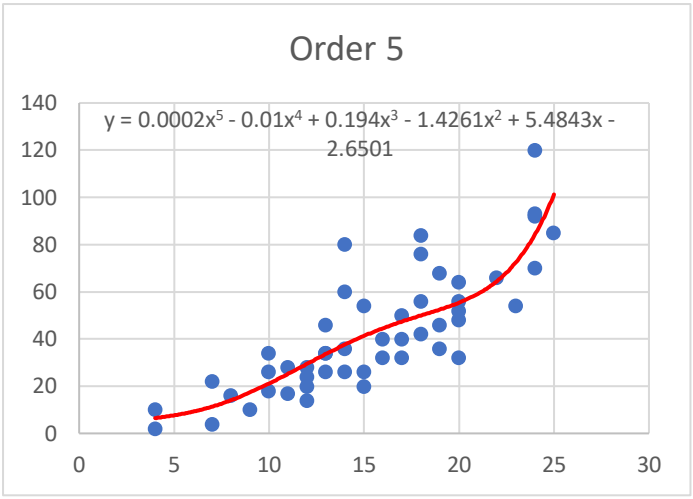
$$f(x) = -7E^{(-05)}x^6 + 0.0061x^5 - 0.2179x^4 + 3.8493x^3 - 35.166x^2 + 157.98x - 260.1$$

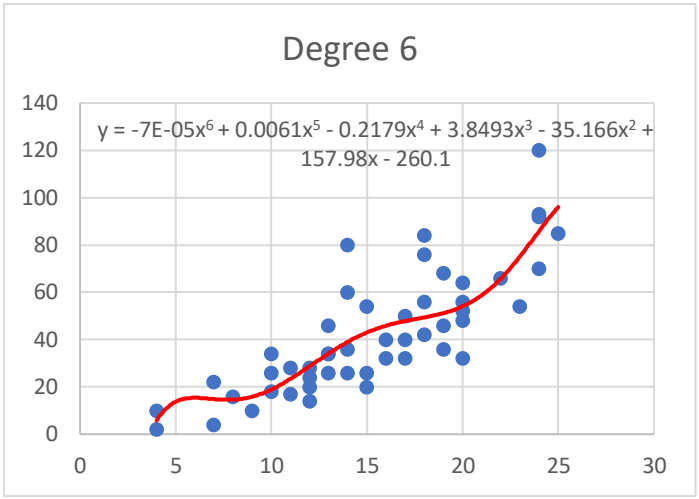




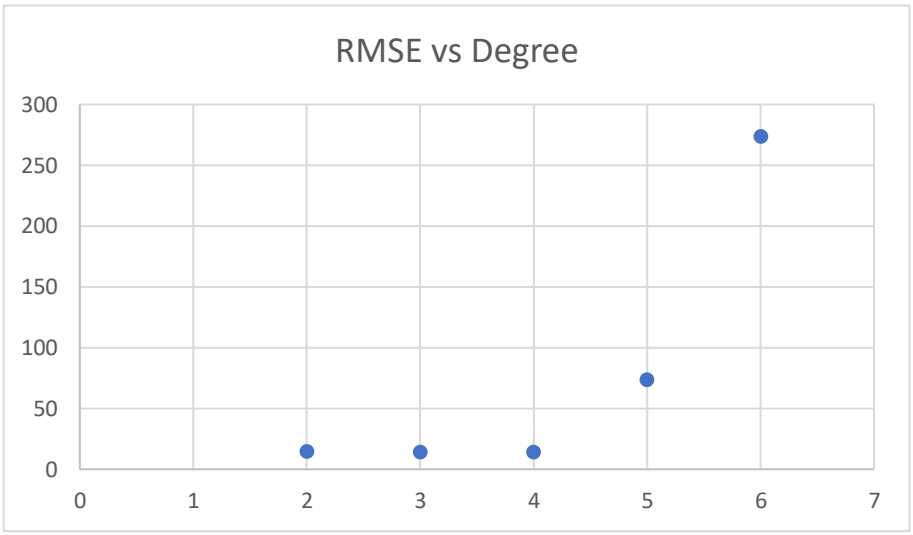








Degree	RMSE
1	15.0689
2	14.71375
3	14.58681
4	14.35212
5	73.99237
6	274.1467



From the above diagrams and data, a polynomial function of degree 4 is giving us the best result.

References

1. Tresp, Volker (2016). Basis Functions. dbs.ifi.lmu.de/Lehre/MaschLernen/SS2016/. Last accessed on 2 April 2021.

2. Lawrence, Neil D (2015). Basis Functions. https://inverseprobability.com/talks/notes/basis-functions.html. Last accessed on 2 April 2021.

3. Adams, Ryan P (2018). Features and Basis Functions. https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/basis-functions.pdf. Last accessed on 2 April 2021.

4. McGill University. Basics of Basis Functions. https://www.psych.mcgill.ca/misc/fda/ex-basis-a1.html. Last accessed on 2 April 2021.

5. Ezekiel, M. (1930). Methods of Correlation Analysis. Wiley.

6. Wikipedia. Wikipedia.org.

7. Marsland, Stephen (2015). Machine Learning: An Algorithmic Perspective (2nd ed.) CRC Press.

8. Notes and other material from the course CS F320 Foundations of Data Science taught at Birla Institute of Technology and Science in Second Semester 2020-21 by Dr. Navneet Goyal.