Approximation by Basis Pursuit: Background and Application to the Construction of Efficient Spline Approximations

Babita Timalsina

Project Submitted to
The Eberly College of Arts and Sciences
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science in Mathematics
Area of Emphasis: Applied Mathematics

Prof. Harvey Diamond, Chair
Adam Halasz, Ph.D.
Robert Mnatsakanov, Ph.D.

Morgantown, West Virginia, USA 2022

Keywords: Basis Pursuit, Signal Processing

Abstract

Approximation by Basis Pursuit: Background and Application to the Construction of Efficient Spline Approximations

Babita Timalsina

Basis Pursuit was developed primarily as a tool in the field of signal processing, beginning in the mid 1990's. The idea is to model the behavior of discrete signals using a wide range of functional behaviors and scales and to obtain an accurate and efficient representation of the signal using a minimal number of functions from a large "dictionary" of possible behaviors. The key observation is by formulating the representation as an ℓ_1 optimization, the problem can be posed as a linear program so that the optimal solution uses no more than the number of constraints - it must be a basic feasible solution.

While the problem has been explored in signal processing, we are here interested in the possible application to approximation of functions as classically considered in analysis. We present a number of applications in approximation with a dictionary consisting of multiresolution cubic spline spaces, with varying objective functions, but optimizations that ultimately must minimize a linear objective function. While signal processing applications are concerned with efficient solution of very large linear programs, here we can limit the sizes of the problems and study the nature of the solutions themselves. We use interpolation, uniform approximation, and formulations involving blended approximation, with objective functions involving ℓ_1 terms blended with uniform or quadratic penalty functions.

Acknowledgements

I would like to thank my advisor, Professor Harvey Diamond, for suggesting this problem and for providing background in the underlying theory and computational implementations. I would also like to thank Dr. Adam Halasz and Dr. Jessica Deshler for constant interaction and support during my graduate study. Finally, I would like to remember to my husband Ghadendra Bhandari for his encouragement.

Table of Contents

List of Figures

CHAPTER 1

Introduction

Function approximation deals with the problem of reconstructing a complex or unknown function, being only provided with a finite set of data points sampled from this function. In some cases we know or can compute the function exactly through its domain and wish to generate an efficient approximation from some family of functions. The best approximation minimizes the difference between the original function and the approximation according to some metric. Function approximations arises in many branches of science and is fundamental to applications such as pattern recognition, prediction and classification. In 1885, Karl Weierstrass published a proof of a theorem, nowadays known in its more general form as the Stone-Weierstrass theorem, which says that any continuous function on the closed interval $[a, b]$ may be arbitrarily closely approximated by polynomials [1]. Much classical approximation involved approximation by polynomials and /or trigonometric polynomials. We have approximated the functions by Basis pursuit methods using spline function.

Basis pursuit is an approach to obtaining a continuous representation of a signal by decomposing it into a superposition of elementary wave forms with sparse coefficients[2]. In basis pursuit, the primary notion is to find a solution that minimizes the ℓ_1 norm of the solution rather than the traditional sum of squares ℓ_2 norm of the residual error[3]. Typically, the larger the dictionary, the better a signal can be approximated. However, not all signals need all of the basis functions for a reasonable approximation. The idea of basis pursuit is to choose a small subset of a larger set, or dictionary, of basis functions that may be over determined. The mathematical optimization form of the basis pursuit is given by[4, 5]

$$\min_x ||x||_1 \text{ subject to } y = Ax \tag{1.1}$$

where, x is a $N \times 1$ solution vectors, y is $M \times 1$ vector of observations,
A is $M \times N$ matrix and $M < N$[6].
Because of the non-differentiability of the ℓ_1 norm, this optimization principle leads to decompositions that can have very different properties from other signal representation ℓ_2 norm. We have discussed basis pursuit method in chapter 2 and its equivalence of linear programming in chapter 3.

The basis pursuit is usually applied in the cases where there is an under determined system of linear equations $y = Ax$ that must be exactly satisfied. The principle of basis pursuit is find a representation of the signal whose coefficients have minimal $\ell 1$ norm. Basis pursuit can be converted to linear programming which is explained in the next chapter. So, the solution of the optimization problem is a basic feasible solution, using columns that form a basis of R^M, hence the name " basis pursuit".

Another classical method of function approximation uses spline spaces, and more recently, multiresolution spline spaces. Here we can consider interpolation, curve fitting, or approximation in the standard mathematical norms of ℓ_1, ℓ_2, and the uniform norm. A spline of degree n is a piecewise polynomial function of degree n in the variable x (the pieces connect continuously at the knots). The values of x where the pieces of polynomial meet are known as knots, denoted by $n+1$ locations $t_1, t_2, t_3, \ldots, t_{n+1}$. It should be non-decreasing $t_i \leq t_{i+1}$. If each knot is separated by the same distance $t_{i+1} - t_i$ the knot vector and the corresponding B-splines are called "uniform". For a given set of data $\{t_i, y_i\}_{i=1}^{n+1}$, we attempt to choose a spline function $s(x)$ with nodes at $\{x_i\}_{i=1}^{n+1}$ in such a way that $||\bar{y} - \bar{s(t)}||^2$ is minimized. We have discussed B-spline methods in chapter 4. Our dictionary in this project uses uniform B-splines at successively halved resolutions. Such spaces are capable of approximation of not just functions values, but their derivatives as well. On the other hand, they are not as open-ended as the dictionaries used in signal processing and the problems can be readily formulated.

We show some of the needed formulations for carrying out basis pursuit: uniform approximation, blended approaches and quadratic formulation. Chapter 5 concludes the project by providing a synopsis of the results.

CHAPTER 2

Optimization Problems using Basis Pursuit

2.1 Introduction

Basis pursuit is a type of nonlinear, or adaptive, approximation method. If one wishes, say, to optimize an approximation using n terms of an infinite series representation, then the terms retained will depend on the function, so that the n-term optimal approximation of a sum of two functions is not the sum of their individual approximations, hence the "nonlinear" descriptive term applied.

In the case of an orthonormal decomposition, such as wavelets in multiresolution settings, or simply Fourier series, an optimal n-term approximation of a function with respect to the ℓ_2 norm consists simply in taking the n terms whose coefficients (in absolute value) are the largest or equivalently, eliminating all coefficients whose absolute value is less than or equal to the n^{th} largest coefficient. As pointed out in Devore's article [7] this in turn is just thresh-holding, a common method of compression or economization of approximations. It is also an approximation method that depends on the function being approximated and so is adaptive, and nonlinear.

In basis pursuit, we choose basis vectors from a "dictionary" of vectors so as to obtain an expansion $\bar{f} = S_n = \sum_{i=1}^{n} a_i \nu_i$ with the idea that $S_k = \sum_{i=1}^{k} a_i \nu_i$ for $k < n$ provides good approximations of f, i.e. $||\bar{(}f) - \sum_{i=1}^{k} a_i \nu_i||$ is small.In other words, we have selected a basis that provides an efficient approximation. A similar idea is to choose k vectors from our dictionary so that $||f - S_k||$ is as small as possible. Our dictionary would normally consist of vectors $\{\nu_i\}$ or functions $\{\phi_i(x)\}$ which are easy to compute, the idea being to replace f by a suitably short expansion. Such expansion would be most useful where f varies on a range of scales, with our expansion presumably including higher-resolution terms where they are needed.

Consider, for instance, orthonormal bases on R^n. One such choice are the standard basis vectors $\{\bar{e}_i\}$ in this context referred to as the "dirac" dictionary. If our vectors $\bar{f}$ have no particular structure, (for instance randomly generated) then this may work reasonably well. But if our vectors $\bar{f}$ are of the form $f_i = f(ih)$ for some smooth function f, we might wish to use a different basis, perhaps a discrete form of the Legendre polynomials.

Consider the following Matlab code, where the built-in "qr" function is used for the orthogonal-triangular decomposition of a matrix A, with the output in Fig 2.1.

```
x=-1:.1:1;x=x';
A=ones(size(x)); for i=1:20, A=[A x.^i];end
help qr
[Q,R]=qr(A); %our basis vectors are in orthogonal matrix Q
f=@(x) 5*x./(1+25*x.^2).^.5;
figure,plot(x,f(x)),grid on
y=f(x);
c=Q'*y
plot(x,y,x,Q(:,1:5)*c(1:5))
```

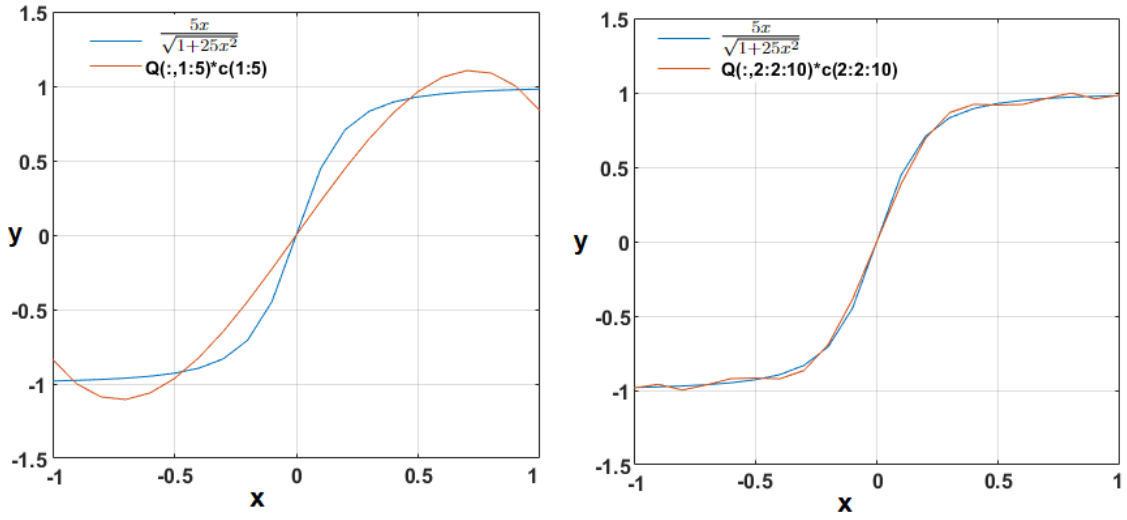


Figure 2.1 Basis Pursuit Illustration.

In the figure Fig:2.1(a) the approximation is produced by the first five vectors

basis vectors. The graph indicates still there is some need of improvement. If we look at the coefficients, all the even "powers" have zero coefficients. In general, we want to use the terms that contribute the most. In this case, it would be 2:2:10 if we want to use the "best five" terms. This approximation shown in figure Fig:2.1(b), which is better than before. Pretty clearly the standard Dirac basis vectors would not have the property that partial sums are good approximations.

So we are left with the following questions: for a given function f, what is a good basis to use? If we look at the sequence of "best partial sums" $\{S(k)\}$ how would that compare if we fixed a value of k and found the best approximation using any k basis functions? We seek methods that are not only nonlinear, but adaptive to the function f under consideration. This is the setting for "basis pursuit".

In basis pursuit, begin with a dictionary of potential basis functions that is over-complete, i.e. we have many more functions than we need to create a basis. These functions will typically be easy to compute with. We use a weighted ℓ_1 norm of the coefficients to effect a selection - as we will see later, this minimization will select a basis for us.

CHAPTER 3

Review of Linear Programming

When we formulate equations modeling an application, they are typically approximated by linear equations because linear equations are so easy to solve. Linear programming is an optimisation problem where the objective function is a linear combination. It has long been known that minimum ℓ_1 optimizations and LP are equivalent[8]. The solution of equation (1.1) can be obtained by solving an equivalent linear program. If A is $m \times n$ with full row rank then there is always an optimal solution with at most m nonzeros variables, called a basic feasible solution. If the nonzero basic variables are $\{x_j\}_{j \in J}$ then $A(:, J)$ is our optimal basis. One can find the solution using simplex method or interior point methods - we do not emphasize algorithms here, however and just use Matlab's linprog.

3.1 Linear Approach

In basis pursuit, begin with a dictionary of potential basis functions that is overcomplete, i.e. we have many more functions than we need to create a basis. These functions will typically be easy to compute with. We use a weighted ℓ_1 norm of the coefficients to effect a selection - as we will see later, this minimization will select a basis for us. We'll consider first interpolation of data.

$$\min \sum w_i |a_i| \ (\{w_i\} \text{ are given positive weights }), \ \sum a_i \phi_i = f$$

where $\{\phi_i\}$ are the functions in our dictionary. This is a linear program that can be put into the standard form.

$$\begin{aligned} \min\ & \bar{c}^T \bar{x} \\ A\bar{x} &= \bar{b} \\ \bar{x} &\geq 0 \end{aligned} \tag{3.1}$$

we replace $a_i = a_i^+ - a_i^-$ with both terms non-negative so that $|a_i| = a_i^+ + a_i^-$. In any optimal solution it is clear that either $a_i^+ = 0$ or $a_i^- = 0$ (or both). The linear program then looks like:

$$\min \sum w_i (a_i^+ + a_i^-)$$

$$\begin{bmatrix} \phi & -\phi \end{bmatrix} \times \begin{bmatrix} \bar{a}^+ \\ \bar{a}^- \end{bmatrix} = \begin{bmatrix} \bar{f} \end{bmatrix}$$

The linear programming formulation ensures that at most n of the coefficients (n =#data values) will be nonzero and this feature is what chooses our basis vectors. In our Matlab implementation we opt to keep a_i^+, a_i^- next to each other in the matrix and vector:

$$\begin{bmatrix} \phi_1(\bar{x}) & -\phi_1(\bar{x}) \vdots\ \phi_2(\bar{x}) & -\phi_2(\bar{x}) \vdots\ \cdots \vdots\ \phi_n(\bar{x}) & -\phi_n(\bar{x}) \end{bmatrix} \times \begin{bmatrix} \bar{a}_1^+ \\ \bar{a}_1^- \\ \bar{a}_2^+ \\ \bar{a}_2^- \\ \vdots \\ \bar{a}_n^+ \\ \bar{a}_n^- \end{bmatrix} \tag{3.2}$$

3.2 Quadratic Programming and other formulations

In some contexts we want to relax the interpolation constraint but "penalize" errors. This can be accomplished by "blending" terms as below in a general formulation.

$\min\ \bar{c}^T \bar{x} + \lambda ||A\bar{x} - \bar{b}||_2^2$

$\min\ \bar{c}^T \bar{x} + \lambda ||\bar{e}||_2^2$, Where, $A\bar{x} - \bar{b} = \bar{e}$

This is a quadratic programming problem and is computationally more difficult than linear programming. The value of λ is chosen by the user to empha-

size/deemphasize the effect of pointwise errors.

The above formulation suggests the following alternatives:

min $\bar{c}^T\bar{x} + \lambda||\bar{e}||_1$, Where, $A\bar{x} - \bar{b} = \bar{e}$

This can be formulated as a linear programming problem with additional variables $e_i = e_i^+ - e_i^-$. The norm $||\bar{e}||_1 = \sum |e_i|$ can be modified with weights as in 3.1. One may, for instance want an approximation where the relative errors are controlled

min $\bar{c}^T\bar{x} + \lambda||\bar{e}||_\infty$

These can be formulated for basis pursuit as before. In each of these problems, given any choice for M or $\bar{e}$, the coefficients of $\{\phi_j\}$ will represents the solution of a linear programming problem, and hence a basis will be chosen from among the $\{\phi_j\}$ as noted before.

(i)

$$\min \sum w_j(a_j^+ + a_j^-) + \lambda||\bar{e}||_2^2$$

$$\left[\phi_1(\bar{x}) \quad -\phi_1(\bar{x}) \vdots\ \phi_2(\bar{x}) \quad -\phi_2(\bar{x}) \vdots\ \cdots \vdots\ \phi_n(\bar{x}) \quad -\phi_n(\bar{x})\right] \times \begin{bmatrix} \bar{a}_1^+ \\ \bar{a}_1^- \\ \bar{a}_2^+ \\ \bar{a}_2^- \\ \vdots \\ \bar{a}_n^+ \\ \bar{a}_n^- \end{bmatrix} - \left[\bar{e}\right] = f \tag{3.3}$$

$a_j^+,\ a_j^- \geq 0,\ j = 0, 1, 2, ...n,\ e_i$ unconstrained. This is the quadratic programming formulation for which we will use Matlab's optimization function *quadprog.m*.

(ii) In a formulation that includes a blended penalty term for $||e||_1$ we would solve

$$\min \sum_{j=1}^{m} w_j(a_j^+ + a_j^-) + \lambda \sum_{j=1}^{m}(e_i^+ + e_i^-)$$

$$\begin{bmatrix}\phi_1(\bar{x}) & -\phi_1(\bar{x}) \vdots\ \phi_2(\bar{x}) & -\phi_2(\bar{x}) \vdots\ \cdots \vdots\ \phi_n(\bar{x}) & -\phi_n(\bar{x})\end{bmatrix} \times \begin{bmatrix}\bar{a}_1^+ \\ \bar{a}_1^- \\ \bar{a}_2^+ \\ \bar{a}_2^- \\ \vdots \\ \bar{a}_n^+ \\ \bar{a}_n^-\end{bmatrix} - \begin{bmatrix}\bar{e}^+\end{bmatrix} + \begin{bmatrix}\bar{e}^-\end{bmatrix} = f \tag{3.4}$$

(iii) The subsequent formulation involves a penalty term using $||e||_\infty$ norm.

$$\min \sum_{j=1}^{m} w_j(a_j^+ + a_j^-) + \lambda M$$

$a_j^+,\ a_j^- \geq 0,\ j = 0, 1, 2, ...n,\ e_i$ unconstrained. $M = ||e||_\infty$ is true for the optimal solution.

$$\begin{bmatrix}\phi_1(\bar{x}) & -\phi_1(\bar{x}) \vdots\ \phi_2(\bar{x}) & -\phi_2(\bar{x}) \vdots\ \cdots \vdots\ \phi_n(\bar{x}) & -\phi_n(\bar{x})\end{bmatrix} \begin{bmatrix}\bar{a}_1^+ \\ \bar{a}_1^- \\ \bar{a}_2^+ \\ \bar{a}_2^- \\ \vdots \\ \bar{a}_n^+ \\ \bar{a}_n^-\end{bmatrix} - f \leq M \tag{3.5}$$

$$-\begin{bmatrix}\phi_1(\bar{x}) & -\phi_1(\bar{x}) \vdots\ \phi_2(\bar{x}) & -\phi_2(\bar{x}) \vdots\ \cdots \vdots\ \phi_n(\bar{x}) & -\phi_n(\bar{x})\end{bmatrix} \begin{bmatrix}\bar{a}_1^+ \\ \bar{a}_1^- \\ \bar{a}_2^+ \\ \bar{a}_2^- \\ \vdots \\ \bar{a}_n^+ \\ \bar{a}_n^-\end{bmatrix} + f \leq M \tag{3.6}$$

(iv) Here we specify a tolerance (call it M) and seek a uniform approximation with

maximum error M. The formulation in iii would apply, except without the M term in the objective function and the value of the variable M specified. This is treated later in section 5.3 and we do not rewrite the entire formulation here.

CHAPTER 4

Multi-resolution Spline Approximations

4.1 Introduction

Formulation of interpolation/approximation problems using basis pursuit and its application to numerical examples with a Matlab implementation have been applied with a multi-resolution spline dictionary. A spline function is a piecewise defined polynomial function with several beneficial properties such as numerical stability of computations, local effects of coefficient changes and built-in smoothness between neighboring polynomial pieces. A common application of spline functions is fitting of data which can be done either by interpolation or approximation of data points. In general, an interpolating function passes through the data points, while an approximating function minimizes the residuals between the function and the data without passing through the data points [9]. Representations using splines are popular in computer-aided design, engineering, modeling and graphics for the drawing of curves, objects and surfaces. In addition, they are for calculating trajectories of computer controlled industrial machines and robots.

A B-spline of degree n is a minimally supported piecewise polynomial function of degree n in the variable x (the pieces connect continuously at the knots). The values of x where the pieces of polynomial meet are known as knots, denoted by $n+2$ locations $t_0, t_1, t_2, t_3, \ldots, t_{n+1}$. It should be non-decreasing $t_i \leq t_{i+1}$. If each knot is separated by the same distance $t_{i+1} - t_i$ the knot vector and the corresponding B-splines are called "uniform". For a given set of data $\{x_i, y_i\}_{i=1}^{n+1}$, we attempt to choose a spline function $s(x)$ with knots at $\{t_i\}$ in such a way that $\{y_i - S(t_i)\}$ is minimized.

4.2 Recursion Relation of B-spline function

B-splines have the property that any spline function of degree n on a given set of knots $\bar{t}$ can be expressed as a linear combination of B-splines

$$S_{n,t}(x) = \sum_i \alpha_i B_{i,n}(x) \tag{4.1}$$

B-splines play the role of basis functions for the spline function space, hence the name. On an interval [a,b] a basis for the spline space is formed by all the B -splines whose support has nontrivial intersection with the interval in question. We begin with the B-splines of degree 0, whose spline spaces have jump discontinuities at the knots.

$$B_{i,0,\bar{t}}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Expressions for B-splines of higher degree polynomial can be derived from the degree-0 B-splines by means of the Cox–de Boor recursion formula[10]

$$B^{(i)}_{i,n,t}(x) = \frac{x - t_i}{t_{i+n} - t_i} B_{i,n-1,t}(x) + \frac{t_{i+1+n} - x}{t_{i+1+n} - t_{i+1}} B_{i+1,n-1,t}(x) \tag{4.3}$$

Where,$B_{i,n,t}(x)$ i^{th} B-spline of degree n with knot sequence $t := \{t_i, ..., t_{i+1}\}$, $t_j \leq t_{j+1}$.

If all knots in a B-spline are distinct, then their derivatives are continuous up to order $n - 1$. The following plot 4.1 shows the basis function $B_{0,n}$ for the knots $x_j = j$[11]. Spline within the knots marked by a straight line is non-zero and outside it is zero. As the degree increases, the B-spline becomes increasingly smooth.

A linear spline basically connects the data points with a linear function and the corresponding B-spline has a tent-shape. In the case of quadratic spline, quadratic functions are used instead of linear function. Cubic splines are most frequently used in applications as they have continuous curvature, and have good approximation order equivalent to local cubic polynomial approximation.

4.3 Cubic B-spline

We use the recursion formula given by equation:(4.3) to produce cubic B-splines centered at x=0 with uniform knot spacing of 1. This B-spline is denoted by

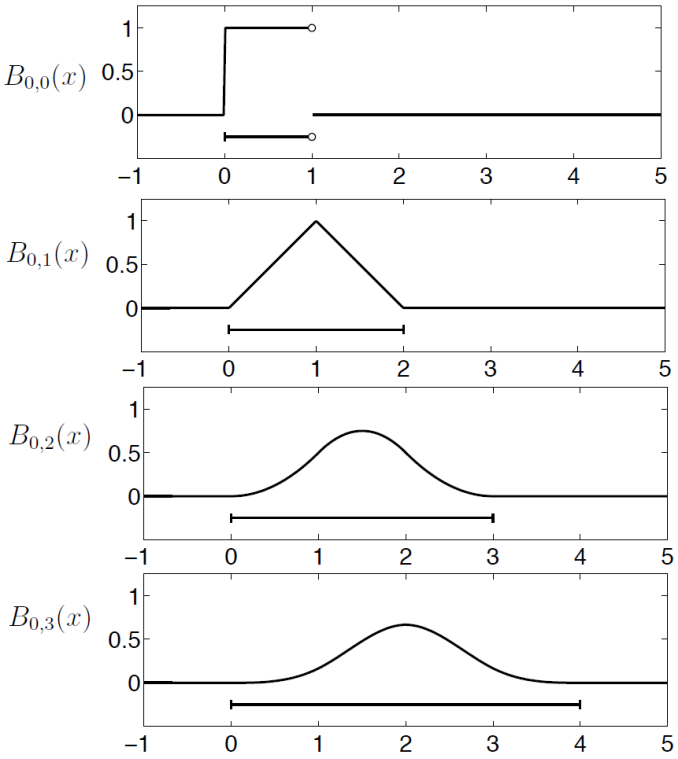


Figure 4.1 B-spline from degree 0 to 3.

$B_3(x)$ as shown in figure(4.2 a). It has non-zero value from -2 to 2 and zero outside. The number of segments required for this spline are 4 and knots are 5. The spline is translated by unit intervals to obtin other B-splines; $B_3(x-1)$; as shown in figure (4.2 b). Using a similar process we can create a large number of spline functions and when we scale the splines in power of two we obtain a multiresolution spline space spanned by scalings and translations of which forms our dictionary.

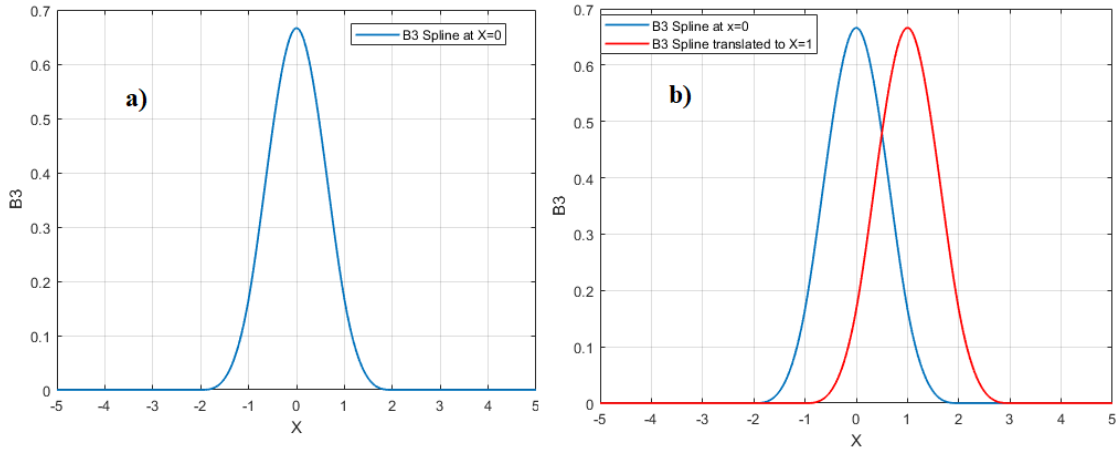


Figure 4.2 B-splines: a) A cubic B-spline centered at x=0, b) Two Cubic B-splines obtained from translation of previous.

CHAPTER 5

Applications and Results

5.1 Basis pursuit involving multi-resolution B-spline interpolation and approximation

Our dictionary for interpolation and approximation consists of a multi resolution spline generated by B-splines $\{B(2^m x - k)\}_{m,k}$, where m specifies the scaling (knot spacing of 2^{-m}) and k identifies translations along the x-axis. Cubic B-spline code is developed from the recursion formula 4.3. Now, as an example let us take $m = 2$ on the interval [-1,1]. Then, $B_3(4x - k)$, comparing with $B_3(x/h - k)$, uses knot spacing h=1/4. Figure 5.1 is plot of a B-spline basis functions $B_3(4x-k) = B_3(4(x-k/4)), k = -5, -4, ..., 4, 5$. The space spline is spanned by the B-splines centered at each of the points k/4 and so there are $9 + 2 = 11$ B-splines. Note that if there is (x, y) data specified at the centers $x : -1, -3/4, .., 3/4, 1$ then this is 9 pieces of data.

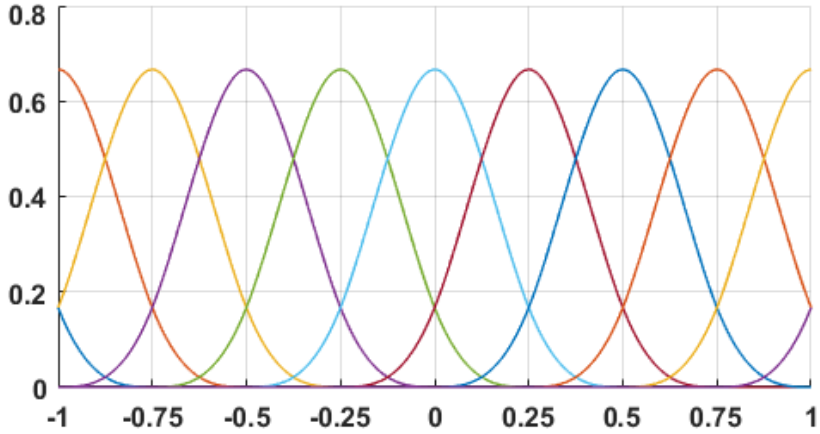


Figure 5.1 A set of B-splines of degree 3 constructed by translation.

We will use this interval [-1,1] for multiple values of m (without loss of generality), viz., m=0,1,2 and 3 (i.e. spacing 1, 1/2, 1/4, 1/8) for our scaling. For example m=0 will involve the B-spline basis $\{B(x-k)\}, k = \{k = -2, -1, 0, 1, 2\}$ with knot spacing of 1 and then this needs 5 set of basis splines. Similarly, it needs 7, 11, 19 set basis splines corresponding to m = 1,2, and 3. And altogether it should be 42 B-spline functions in our dictionary. Firstly, we will show some of the needed formulations for carrying out basis pursuit, then we will move to couple of other problems: uniform approximation, blended approaches and quadratic formulation.

5.2 Basis pursuit interpolation

This is our first model, where we interpolate the data and use basis pursuit to find an efficient representation of the interpolant. The input data consists of data points $\{x, y\}$ where x = points in [-1,1] with spacing h defined above and our spline interpolant given by

$$s(x) = \sum_{i=0}^{imax} \sum_{k=-2^i-1}^{2^i+1} c_k B(2^i x - k) \tag{5.1}$$

Let $t(k)$ is the array center and spaced by 2^{-i}. We have the constraints

$$min \sum 2^i |c(i,j)|$$

To these constraints we add the interpolation constraints

$$s(x) = \sum_{i=0}^{imax} \sum_{k=-2^i-1}^{2^i+1} c_k B(2^i(x - t(k)) = f(x) \text{at } x = x(j) \tag{5.2}$$

Here, 2^i works as a penalty coefficient in the objective function, so that higher resolution splines are penalized. Equation 5.2 gives matrix A which has 42 columns as discussed above. To interpolate the discrete data (x, g(x))

$$\sum_{i,j} c(i,j) B(2^i(x - t_{i,j})) = g(x)$$

with (i,j) sufficient for interpolation. In general the equations will be undetermined by the interpolation conditions. This is our basis pursuit formulation of interpolation with many solutions. The linear program will "select" an optimal basis for us. Note that $c(i,j) = c_{i,j}$ is the coefficient of a centered B-spline whose support is scaled by

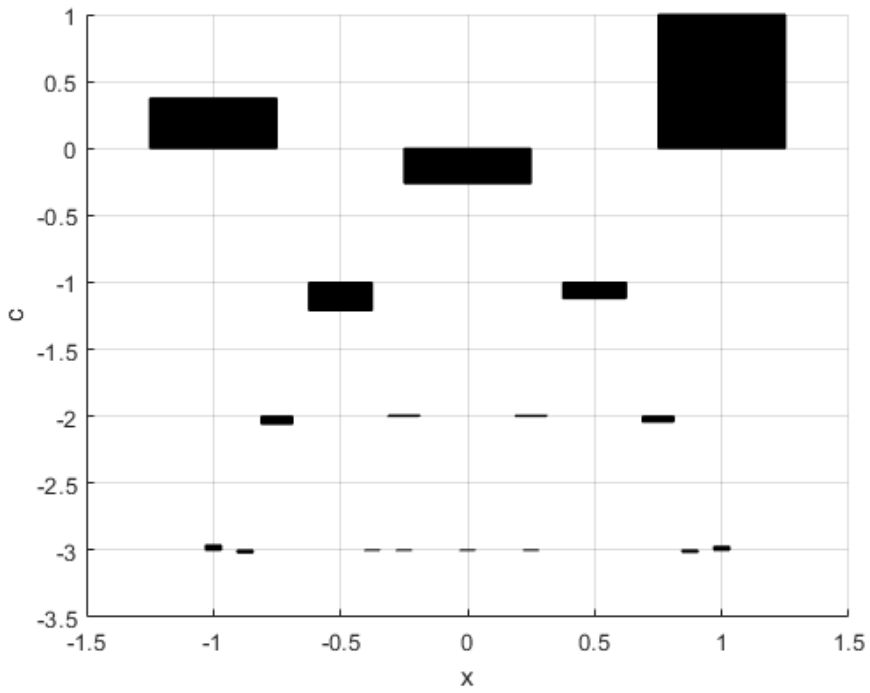


Figure 5.2 Schematic representation of coefficient's size and center.

2^{-i}, with the values of j chosen so as to include all B-splines whose support intersects [-1, 1] in a non-trivial interval. For each value of i, our splines are $B((2^i(x-t))$ with the centers at the elements of t. For cubic splines we need one "extra" spline to the left and one to the right of the interval [-1,1] in order to span the associated spline space. The array $t(k)$ is $t = -1 - w * (2^{-i}) : 2^{-i} : 1 + w * (2^{-i})$; where offset w=1. Then, we have to formulate the problem for linear programming optimization. As discussed in linear approach in 3.2, we developed another matrix by duplicating each columns and multiplying by -1, ie, A:{Aj, -Aj}. We can calculate the basis pursuit solution to get coefficient matrix {c} using in built function "*linprog*" in Matlab: $c = linprog(f, [\,], [\,], Aeq, beq, lb, ub)$. Where, Aeq=A and beq= y. It requires that the user define a set of lower and upper bounds on the design variables, x, so that the solution is always in the range $0 = lb \leq x \leq ub = \infty$. Next, we reform our variables (the coefficients: c) so that they are exclusively non-negative. $c_{ij} = c_{ij}^{+} - c_{ij}^{-}$ and $|c_{ij}| = c_{ij}^{+} + c_{ij}^{-}$.

Figure 5.2 shows which 17 of the 42 B-splines are used. The height of the rectangle (Δy) gives the relative size of the coefficient with the width proportional to the knot spacing and the base drawn at $y = -1$ when $h = 2^{-i}$ is the knot

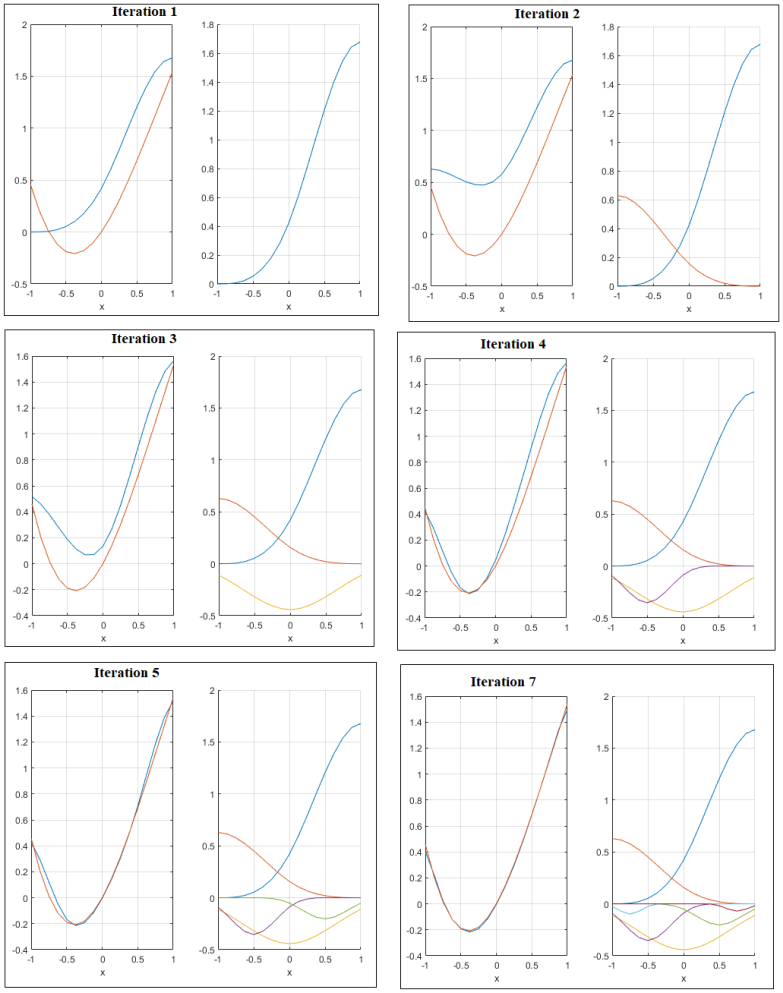
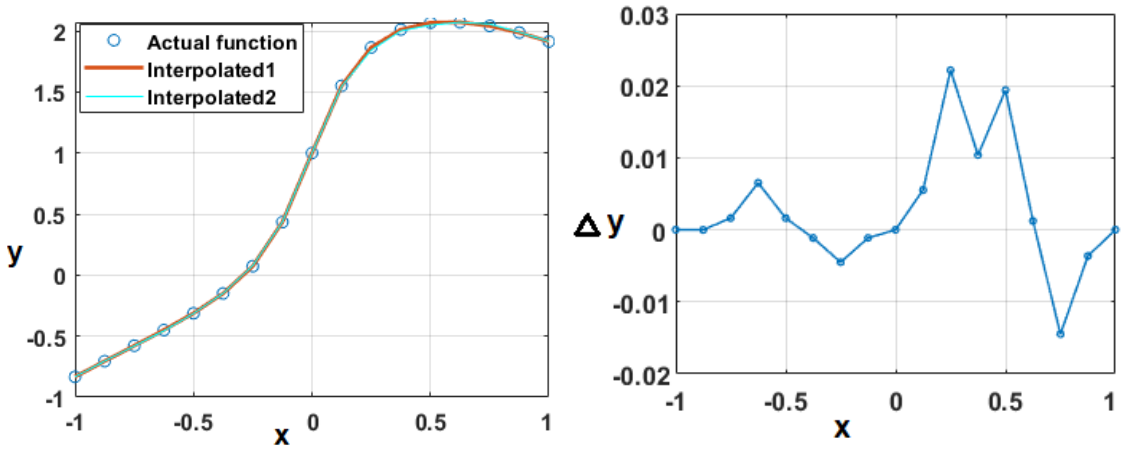


Figure 5.3 A sequence of operation for interpolation the function $y = xcos(x) + x^2$. In the left graphs of each iteration red line denotes actual function and blue line denotes approximated function.

spacing. If the coefficients are ordered by size, we obtain the sequence of partial approximations shown below, including the first 7 largest terms. We interpolated the function $y = xcos(x) + x^2$ and shown in figure 5.3 at the points $x = -1 : 1/8 : 1$ and used basis pursuit to find an efficient representation of the approximation.

We also interpolated another function $g(x) = tan^{-1}(5x) + cos(x)$. The interpolated function and actual function has been shown in figure 5.4(a). There are 17 nonzero coefficients but basis pursuit has economized the coefficients of the high resolution splines to be small. Thresholding these values can further economize the representation and the figure on the right is the residue when coefficients below 0.05 in absolute value are thresholded, leaving 11 nonzero coefficients in the



(a) (b)

Figure 5.4 (a)Actual function $g(x) = tan^{-1}(5x) + cos(x)$ and interpolated data (b) Residue

representation.

5.3 Uniform Approximation

In this case, we solved

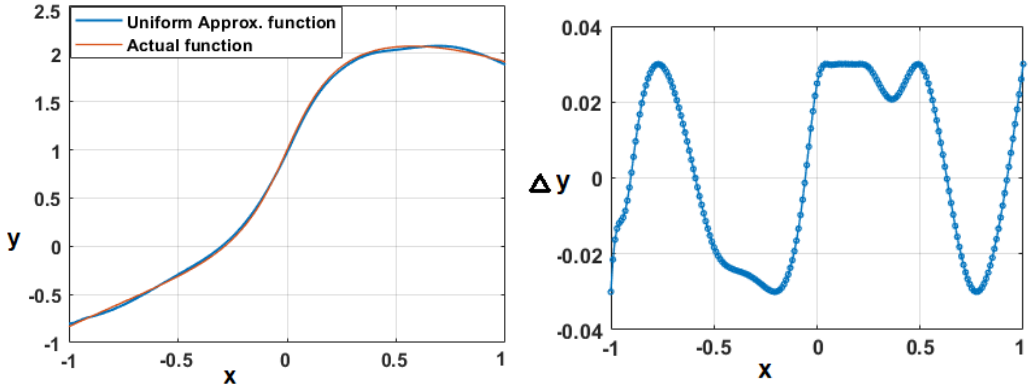
$$\min \sum 2^i |c(i,j)| \text{ such that } |\sum_{i,j} c_{ij} B(2^i(x - t_{i,j})) - g(x)| \leq M$$

where the value of M is specified according to the desired tolerance. The vector x is a discrete sample of the domain but will be a good proxy in practice for the points of interest, e.g. x = -1:0.01:1 for the interval [-1,1]. Note that if desired, it is a simple matter to modify the problem so as to obtain a uniform relative error. The value of chosen for M will of course need to be feasible in light of the dictionary used, but on the other hand, for a continuous function, there is always a dictionary for any positive M that leads to a feasible solution, simply by taking a fine enough resolution in our space.

It is simple matter to note that $|Ax - b| \leq M$ is equivalent to $Ax \leq M + b, -Ax \leq M - b$ so from the matrix AA in the prior section, wherein we split the coefficients into positive and negative parts to form the composite matrix

$$AA = \begin{bmatrix} AA \\ -AA \end{bmatrix}$$

We have approximated the same function $g(x) = tan^{-1}(5x) + cos(x)$. The uniform approximation and actual function are shown in figure 5.5(a) and the residue has been plotted in figure 5.5(b). Our tolerance was $M = 0.03$ and there were only 11 nonzero coefficients in the solution.



(a) (b)

Figure 5.5 (a)Actual function $g(x) = tan^{-1}(5x) + cos(x)$ and uniform approximated data (b) Residue

5.4 Blended approaches

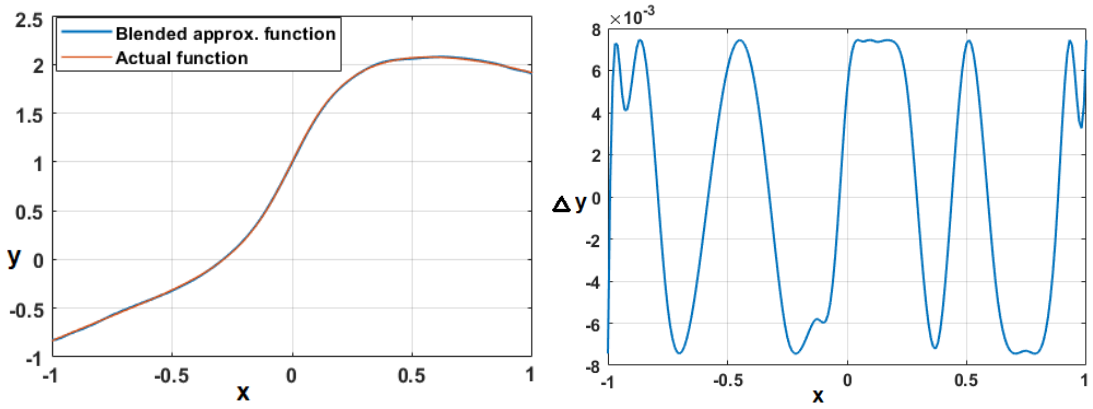
Here, we used a combination of our ℓ_1 objective $\sum 2^i|c(i,j)|$ for the coefficients and the uniform ℓ_∞ norm M of the error.

$$min(M + \lambda \sum 2^i|c(i,j)|)$$

such that

$$|\sum_{i,j} c(i,j)B(2^i(x - t_{i,j})) - g(x)| \leq M$$

Here, we need to incorporate into the constrains as a variable, and into the objective function. This requires an augmented column in our matrix and in the objective function vector. We are approximating the same function $g(x) = tan^{-1}(5x) + cos(x)$. The blended approximated function and actual function has been shown in figure 5.6(a) and the residue has been plotted in figure 5.6(b). The value of λ used here is 0.1. Following list shows the fifteen non-zero value of coefficients c.



(a) (b)

Figure 5.6 (a)Actual $g(x) = tan^{-1}(5x) + cos(x)$ and blended approximated functions (b) Residue

-0.83356
2.534487
-0.056939
0.22383
1.078316
-0.28268
0.21265
0.643833
0.15098
0.186599
-0.03157
0.00605
-0.06529
0.009861
0.04926

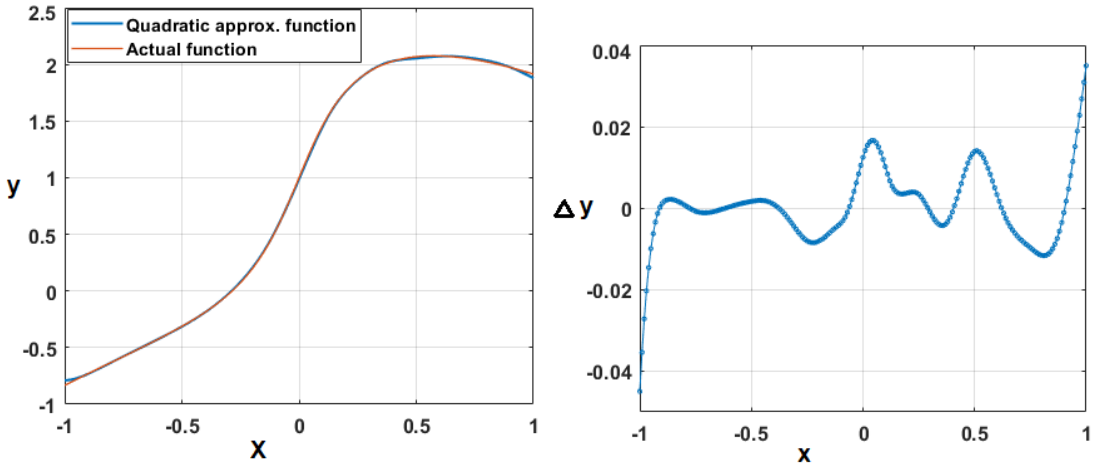
5.5 Quadratic programming formulation

Finally, we used a quadratic programming formulation that is seen in the literature, employing an ℓ_2 norm on the error

$$min\lambda \sum 2^i |c(i,j)| + \frac{1}{2} \sum e(x)^2$$

$$\sum_{i,j} c(i,j) B(2^i(x - t_{i,j})) - g(x) = e(x)$$

We used inbuilt Matlab function "quadprog" as $x = quadprog(H, f, A, b, Aeq, beq, LB, UB)$, where H represents a quadratic term $\frac{1}{2}x^T Hx$ so that $H = I$ in our case. The initial setup is the as in the previous blended approach. The approximated function according to quadratic formulation and actual function has been shown in figure 5.7(a) and the residue has been plotted in figure 5.7(b). The value of λ used here is 0.1, and there are twelve nonzero coefficients.



(a) (b)

Figure 5.7 (a)Actual $g(x) = tan^{-1}(5x) + cos(x)$ and quadratic approximated functions (b) Residue

5.6 Discussion

We were successful in using basis pursuit in several formulations to approximate functions from multi-resolution spline spaces. There are many parameter choices made along the way that could be adjusted and the results studied in similar fashion. Each of the methods used was effective in picking out an efficient basis.

Uniform approximation or a blended uniform approach are more "faithful" to the linear formulation than the more customary blended quadratic approach and seem to do just as well. If the idea is to obtain an efficient approximation with few nonzero coefficients, then interpolation is probably not the best approach, as it naturally will use as many functions as there are equality constraints.

These problems are a jumping off point for further study. Some questions might be: are there any desirable optimality properties we can associate with basis pursuit solutions of this sort, and which formulations are more desirable in which contexts? Also, with a linear programming formulation, a variety of other issues can be included, e.g. positive approximations for positive functions, issues of shape preservation, or uni-directional approximations. These can be naturally incorporated into the constraints, whereas classical approximation approaches, such as least squares, are indifferent at best to such questions.

Bibliography

[1] Allan Pinkus. Weierstrass and approximation theory. *Journal of Approximation Theory*, 107(1):1–66, 2000.

[2] Chen, Scott S. Donoho, David L. Saunders, Michael A. Atomic decomposition by basis pursuit. *SIAM Review*, 43:129–159, 2001.

[3] Patrice Simard and Jérôme Antoni. Acoustic source identification: Experimenting the ℓ_1 minimization approach. *Applied Acoustics*, 74(7):974–986, 2013.

[4] Andreas M. Tillmann. Equivalence of Linear Programming and Basis Pursuit. *Proc. Appl. Math. Mech.*, 15:735–738, 2015.

[5] Lorenz, Dirk A. Pfetch, Marc E. Tillmann, Andreas M. Solving basis pursuit: Heuristic optimality check and solver comparison. *ACM Transactions on Mathematical Software*, 41:1–29, 2015.

[6] Mallat Stéphane. *A Wavelet Tour of Signal Processing The Sparse Way*. Sciencedirect, 2009.

[7] Ronald A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

[8] W. Bloomfield, P. Steiger. *Least Absolute Deviations:Theory, Applications, and Algorithms,*. Springer, 1983.

[9] Jorge Holling, Klause Horner. *Approximation and Modelling with B-Splines*. SIAM, 2013.

[10] Knut Lyche, Tom Morken. *Spline Methods Draft*. 2008.

[11] Julian Valentin. Flavors and types of b-splines. 2019.