



TAMPERE UNIVERSITY OF TECHNOLOGY

MARKKU MÄKITALO

Optimization of polynomial and rational variance-stabilizing transformations

Master of Science Thesis

Examiner: Alessandro Foi
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 6th May 2015

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

MARKKU MÄKITALO: Optimization of polynomial and rational variance-stabilizing transformations

Master of Science Thesis, 62 pages

May 2015

Major: Signal Processing

Examiner: Alessandro Foi

Keywords: variance stabilization, optimization, Poisson, Poisson-Gaussian, Rice, Rayleigh, interpolation, approximation

Digital imaging devices and other similar data acquisition devices are subject to various errors, some of which are signal-dependent and others signal-independent. An important part of processing such data is to properly model the random noise, in other words the difference between the measured data and its expectation. For instance, the raw data of imaging sensors can be modelled via a mixed Poisson-Gaussian distribution, and the noise in magnetic resonance images with a Rice distribution.

One common method of dealing with such heteroskedastic data (i.e., data with signal-dependent noise variance) is to process it with a variance-stabilizing transformation (VST). This renders the noise variance constant throughout the data, thus removing the data-dependency of the noise. The stabilized homoskedastic data can then be processed assuming the noise is signal-independent, typically additive white Gaussian noise. Traditionally used forward VSTs are designed in order to stabilize the data asymptotically. Such a design principle usually implies subpar stabilization accuracy for low-intensity data, which often leads to mediocre performance in practical applications.

In this thesis, we first review the theory on function interpolation and approximation. Then we describe and implement a general modular framework for optimizing parametric VSTs, in particular considering the optimization of polynomial VSTs and rational VSTs. This is done by minimizing a cost functional that measures the discrepancy between the stabilized variance and a constant target function. We present example results for the Poisson-Gaussian and Rice distributions, which demonstrate that the optimized rational VSTs consistently provide more accurate stabilization than the traditional VSTs, whereas the advantage of using optimized polynomial VSTs is less clear. Our framework is also extensible to other noise distributions and other types of VSTs.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO

Tietotekniikan koulutusohjelma

MARKKU MÄKITALO: Varianssia stabiloivien polynomien ja rationaalifunktioiden optimointi

Diplomityö, 62 sivua

Toukokuu 2015

Pääaine: Signaalinkäsittely

Tarkastajat: Alessandro Foi

Avainsanat: varianssin stabilointi, optimointi, Poisson, Poisson-Gaussinen, Rice, Rayleigh, interpolointi, approksimointi

Digitaaliseen kuvantamisprosessiin liittyy sekä signaalista riippuvia että siitä riippumattomia häiriöitä. Mitatun häiriöisen datan ja sen odotusarvon erotuksen, toisin sanoen kohinan, huolellinen mallintaminen on tärkeä osa tätä prosessia. Esimerkiksi digitaalikameroiden tallentamaan raakadataan liittyvää kohinaa voidaan mallintaa Poisson-Gaussisella jakaumalla, ja magneettikuvausdatan kohinaa Rice-jakaumalla. Heteroskedastista dataa, jossa kohinan varianssi riippuu mitatun signaalin arvoista, voidaan käsitellä esimerkiksi varianssia stabiloivan muunnoksen eli VST:n avulla. Tällöin kohinan varianssi saa (ihannetapauksessa) vakioarvon, joten se ei enää riipu mitatusta datasta. Stabiloitua homoskedastista dataa voidaan siis tämän jälkeen käsitellä olettaen, että kohina on datasta riippumatonta, tyypillisesti additiivista normaalijakautunutta kohinaa. Perinteiset muunnokset on suunniteltu asympotoottisesti, jolloin stabilointi on yleensä heikohkoa matalaintensiteettiselle (eli pienen signaali-kohinasuhteen omaavalle) datalle. Tämä heijastuu usein huonoina tuloksina käytännön sovelluksissa.

Tässä diplomityössä käydään läpi epälineaarisen interpoloinnin ja approksimoinnin teoriaa, jonka jälkeen työssä toteutetaan parametrusten VST-muunnosten optimointiin tarkoitettu modulaarinen järjestelmä. Käytännössä tämä tehdään minimoimalla kustannusfunktiota, joka kuvaa stabiloidun varianssin ja vakioarvoisen tavoitevarienssin eroa. Työ keskittyy polynomien ja rationaalifunktioiden optimointiin Poisson-Gaussiselle jakaumalle sekä Rice-jakaumalle, mutta järjestelmä on laajennettavissa myös muille jakaumille ja muuntotyypisille funktioille. Esimerkkitulokset osoittavat, että optimoidut rationaalifunktiot stabiloivat varianssia perinteisiä VST-muunnoksia paremmin, kun taas optimoitujen polynomien tehokkuus on jokseenkin kyseenalainen.

FOREWORD

This thesis is based on some of the research performed by the author at the Department of Signal Processing, Tampere University of Technology, Finland, between 2012 and 2014.

I wish to thank Dr. Alessandro Foi for providing me with the opportunity to do this research and for supervising this thesis. Already prior to this thesis, I have worked with him on various other topics related to variance stabilization, which culminated in the completion of my doctoral thesis in 2013.

My original professional background being in mathematics instead of signal processing has lead me to explore topics related to the latter in such an extent that even after completing my doctoral thesis, it was reasonable to structure my extra studies into another master's degree. Hence, it also gave rise to this master's thesis, which complements some of the topics discussed in my doctoral thesis.

Tampere, 20th May 2015

Markku Mäkitalo

CONTENTS

1. Introduction	1
2. Function interpolation and approximation	3
2.1 Polynomial interpolation	4
2.1.1 Polynomial interpolation as a matrix inversion problem	5
2.1.2 Lagrange polynomial	6
2.1.3 Non-Lagrangian forms	8
2.1.4 Polynomial interpolation error	11
2.1.5 Chebyshev nodes	15
2.1.6 Spline interpolation	18
2.2 Rational function interpolation	24
2.3 Function approximation	28
2.3.1 Minimax approximation	29
2.3.2 Least squares approximation	29
2.3.3 Rational approximation	31
3. Noise models	33
3.1 Gaussian noise	33
3.2 Poisson noise	35
3.3 Poisson-Gaussian noise	36
3.4 Rice and Rayleigh noise	37
3.5 Clipping	39
4. Variance stabilization	41
4.1 Development of VSTs	41
4.2 Poisson and Poisson-Gaussian noise	42
4.3 Rice and Rayleigh noise	44
4.4 Inverse transformations	46
5. Optimization of VSTs using polynomials	47
6. Optimization of VSTs using rational functions	50
6.1 Poisson-Gaussian noise	50
6.2 Rice noise	52
7. Experimental results	54
7.1 Implementation details	54
7.2 Poisson-Gaussian VSTs	55
7.3 Rice VSTs	56
8. Conclusions	62
References	63

TERMS AND THEIR DEFINITIONS

AWGN	Additive White Gaussian Noise
CDF	Cumulative Distribution Function
GAT	Generalized Anscombe Transformation
MRI	Magnetic Resonance Imaging
PDF	Probability Density Function
VST	Variance-Stabilizing Transformation
$p(x), p_n(x)$	Polynomial function (of degree n , unless stated otherwise)
p_i	Polynomial coefficient
$g(x)$	Any target function to be approximated
$g'(x), \frac{d}{dx}(x)$	Derivative of g with respect to x
$g^{(n)}(x)$	n -th derivative of g (with respect to x)
$g \in \mathcal{C}[a, b]$	Function g is continuous in the interval $[a, b]$
$g \in \mathcal{C}^n[a, b]$	Function g is n times continuously differentiable in $[a, b]$
x_i	Interpolation node
y	Noise-free data (intensity values of the ideal unknown image)
z	Data y corrupted by noise (the observed data)
y_i, z_i	Noise-free pixel and noisy pixel, respectively
$E\{z\}$	Expected value of z
$E\{z y\}$	Expected value of z , conditioned by y
$\text{std}\{z\}$	Standard deviation of z
$\text{var}\{z\}$	Variance of z
$P(z)$	Probability distribution of z

μ	Mean value of a Gaussian distribution
σ	Standard deviation of a Gaussian distribution
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and standard deviation σ
$\eta_i \sim \mathcal{N}(\mu, \sigma^2)$	Random Gaussian variable η_i is distributed according to $\mathcal{N}(\mu, \sigma^2)$
a, b	Parameters of an affine-variance Gaussian distribution $\mathcal{N}(0, ay + b)$
ρ_i	Random Poisson variable
$\mathcal{P}(y_i)$	Poisson distribution with mean (and variance) y_i
α	Positive scaling factor of a Poisson-Gaussian distribution; gain
ν	Distance parameter for a Rice distribution
$\mathcal{R}(\nu, \sigma)$	Rice distribution with parameters ν and σ
f	Forward variance-stabilizing transformation
f_θ	Forward variance-stabilizing transformation with parameter(s) θ ; e.g., $f_{\alpha, \mu, \sigma}$

1. INTRODUCTION

Digital imaging devices and other similar data acquisition devices are subject to various errors, some of which are signal-dependent and others signal-independent. An important part of processing such data is to properly model the random noise, in other words the difference between the measured data and its expectation. For instance, the raw data of typical imaging sensors can be modelled via a mixed Poisson-Gaussian distribution, with clipping to account for the limited dynamic range of the sensor [1]. Similarly, the noise in magnetic resonance images can be modelled with a Rice distribution [2].

One common method of dealing with such heteroskedastic data (i.e., data with signal-dependent noise variance) is to process it with a variance-stabilizing transformation (VST). This renders the noise variance (ideally) constant throughout the data, thus removing the data-dependency of the noise; the stabilized data is homoskedastic. In practical applications, such as image denoising, this is typically a three-step procedure: First, the variance is stabilized using a VST. Then, the stabilized data can be denoised with an algorithm designed for additive white Gaussian noise. Finally, a suitably designed inverse [3] of the VST is applied in order to return the processed data to the original range. This approach is not restricted to images, but it can be used with data of any dimension, such as 1-D data, or volumetric data.

Traditionally used forward VSTs, such as the Anscombe transformation [4] for Poisson noise, the generalized Anscombe transformation [5] for Poisson-Gaussian noise, and the asymptotic VST [6] for Rice noise, are designed in order to stabilize the data asymptotically. Such a design principle usually implies that their stabilization accuracy is not particularly good for low-intensity data (i.e., for data with a low signal-to-noise ratio), which may in turn lead to mediocre performance in practical applications [7; 8; 9]. Hence, there is a need for VSTs that are more accurate, yet still simple in form and convenient to use.

In this thesis, we present a general framework for optimizing parametric VSTs. In particular, we implement the optimization of polynomial VSTs and rational¹ VSTs. We discuss the Poisson, Poisson-Gaussian, Rice and Rayleigh distributions, but our modular framework can be extended to encompass other distributions and other types of VSTs. We adopt the approach first presented in [10; 11] for optimizing non-

¹A rational function is a ratio of two polynomials.

parametric VSTs. Specifically, we formulate the problem as an explicit optimization problem, minimizing a cost functional that measures the discrepancy between the stabilized variance and a target function, which in this case is a constant function with a known value. We first introduced rational Poisson-Gaussian VSTs in [12] in the context of using variance stabilization for noise parameter estimation, as we needed to mitigate the inaccurate stabilization of the generalized Anscombe transformation. However, this thesis provides a more general framework with a more detailed description of the optimization procedure.

This thesis is structured as follows. In Chapter 2, we provide a detailed review of some of the theory related to function interpolation and approximation, in particular considering polynomials, splines and rational functions. In Chapter 3, we present the relevant Poisson, Poisson-Gaussian, Rice and Rayleigh noise models, and discuss clipping. Chapter 4 explores variance stabilization and the traditional asymptotic VSTs for these noise models, and elaborates on the importance of a properly designed inverse VST. Chapter 5 introduces our framework for optimizing polynomial VSTs, and Chapter 6 extends it to consider rational VSTs. Practical implementation details and example results are provided in Chapter 7, and finally Chapter 8 presents the conclusions to the thesis.

2. FUNCTION INTERPOLATION AND APPROXIMATION

Generally speaking, the goal of function approximation is to find a function that satisfies some pre-determined conditions: we typically require the function to belong to a certain class of functions (e.g., polynomials, splines, or rational functions), and we want it to be a good approximation of the target function. The goodness of the approximation is dependent on the criterion, with which we choose to measure the difference between the approximating function and the target function; for instance, we may be content with the common least squares error criterion, or we may give more weight to certain function values, whose accuracy is the most critical regarding our application.

The target function may belong to a more general class of functions, and we may not even have explicit knowledge about the form of the target function, but merely know the function values at some locations. In that case, function approximation is also called curve fitting, as we are indeed fitting a certain type of curve to a discrete set of data points.

Concerning the terminology, function approximation and interpolation are sometimes used interchangeably depending on the field of study or application. In this thesis, approximation is used as a general term as described above, and interpolation is used when the approximating function is specifically required to coincide with a prescribed set of points, as defined in Section 2.1.

In this chapter, we will first review some of the methods and theories related to polynomial interpolation (Section 2.1). In particular, we discuss polynomial interpolation as a matrix inversion problem, the Lagrange form of the solution polynomial, and some non-Lagrangian forms of the same polynomial. Then, polynomial interpolation error is analyzed, which serves as a motivation for introducing Chebyshev nodes and spline interpolation. In Section 2.2, we introduce rational functions and discuss rational interpolation. Finally, in Section 2.3 we relax the requirements of exactly matching certain data points, discuss function approximation and the measures of goodness of the approximation, and contemplate the advantages and disadvantages of using rational functions over polynomials.

2.1 Polynomial interpolation

Let us first define what we mean by polynomial interpolation. Say we have a (finite¹) set of $n + 1$ data points (x_i, y_i) , $i = 0, \dots, n$, which adhere to some underlying target function g , and $x_i \neq x_j \forall i \neq j$. In other words,

$$g(x_i) = y_i \quad \forall i = 0, \dots, n.$$

Then, our goal is to find a polynomial

$$p(x) := \sum_{i=0}^n p_i x^i = p_0 + p_1 x + p_2 x^2 + \dots + p_n x^n, \quad (2.1)$$

for which

$$p(x_i) = y_i \quad \forall i = 0, \dots, n. \quad (2.2)$$

Such a polynomial p is said to interpolate g , or be the *interpolant* of g , and the evaluation points x_i are called *nodes*. Note that we use $n + 1$ nodes instead of n for notational purposes, as is traditionally done in the literature; then we can conveniently handle polynomials of degree n instead of degree $n - 1$.

The Weierstrass approximation theorem states that if g is continuous on a closed interval, it can be uniformly approximated by a polynomial to an arbitrary precision (see [13, p. 3] and the references therein). However, the theorem does not provide a means to construct such a polynomial, and furthermore, it does not necessarily apply if we force the polynomial to coincide with a set of values of the function g [14, p. 324] (i.e., in the interpolation case).

On the other hand, regardless of the form of g , a unique polynomial p of at most degree n that satisfies (2.2) is guaranteed to exist by the unisolvence theorem. One proof of this theorem is presented in Section 2.1.1, and alternative proofs can be found in [15, p. 132].

However, both of the above theorems may still imply a very high-degree approximating polynomial, depending on the complexity of g or the number of data points (x_i, y_i) at our disposal. We must also keep in mind that in practical applications the data is often noisy, in which case an exact or near-exact fit using a high-degree interpolating polynomial is typically less useful than regressing to a lower-degree polynomial model. Such practical issues related to function *approximation* are discussed in Section 2.3, but first we build up the theoretical background by reviewing the fundamentals of *interpolation* theory, assuming an exact fit (2.2).

¹For discussion on interpolation with infinitely many conditions, we refer to [16, Ch. V].

2.1.1 Polynomial interpolation as a matrix inversion problem

The requirements (2.2) form a system of $n + 1$ equations, which we can equivalently write in matrix form:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} & x_{n-1}^n \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} & x_n^n \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix},$$

or

$$Vp = y \tag{2.3}$$

in short, where $p = [p_0, p_1, p_2, \dots, p_{n-1}, p_n]^T$ and $y = [y_0, y_1, y_2, \dots, y_{n-1}, y_n]^T$. The $(n + 1) \times (n + 1)$ matrix V in (2.3) is commonly known as the Vandermonde matrix. In general, an $m \times n$ Vandermonde matrix

$$V^{m \times n} = \begin{bmatrix} 1 & v_1 & v_1^2 & \dots & v_1^{n-1} \\ 1 & v_2 & v_2^2 & \dots & v_2^{n-1} \\ 1 & v_3 & v_3^2 & \dots & v_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_m & v_m^2 & \dots & v_m^{n-1} \end{bmatrix}, \tag{2.4}$$

in other words $V_{i,j}^{m \times n} = v_i^{j-1}$, is an efficient tool for evaluating a polynomial of degree $n - 1$ at m points v_1, v_2, \dots, v_m . However, the special case of a square Vandermonde matrix V (2.3) is more relevant to our interests in this section, since that allows us to utilize its inverse matrix V^{-1} without resorting to pseudoinverses.

Let us prove that the inverse matrix V^{-1} does actually exist. According to the Leibniz formula, the determinant of an arbitrary $(n + 1) \times (n + 1)$ square matrix $B = b_{i,j}$ equals

$$\det(B) = \sum_{s \in S_n} \text{sgn}(s) \prod_{i=0}^n b_{i,s_i}, \tag{2.5}$$

where S_n is the group of all permutations of the set of indices $\{0, 1, \dots, n - 1, n\}$, and the sign function $\text{sgn}(s)$ indicates the parity of permutation s ($\text{sgn}(s) = 1$ for an even permutation, and $\text{sgn}(s) = -1$ for an odd permutation). For more details on permutations and their parity, we refer the reader to [17, p. 143]. For the

Vandermonde matrix (2.3), equation (2.5) can be written as [18, p. 3]

$$\det(V) = \sum_{s \in S_n} \operatorname{sgn}(s) \prod_{i=0}^n x_i^{s_i-1} = \prod_{0 \leq i < j \leq n} (x_j - x_i). \quad (2.6)$$

Most importantly, taking into account that we have defined $x_i \neq x_j \forall i \neq j$, equation (2.6) implies that $\det(V) \neq 0$. In other words, the Vandermonde matrix V is non-singular and, thus, its inverse matrix V^{-1} exists.

The above affirms that formulating the polynomial approximation problem (2.2) as a matrix inversion problem (2.3) is sensible, and proves that there exists a unique solution (as anticipated by the unisolvence theorem). That is, determining the polynomial coefficients p_i , for which

$$p(x) = \sum_{i=0}^n p_i x^i = y_i \quad \forall i = 0, \dots, n,$$

is equivalent to solving (2.3) for the coefficient vector p :

$$p = V^{-1}y. \quad (2.7)$$

Section 2.1.2 discusses how the solution can be explicitly constructed.

2.1.2 Lagrange polynomial

In practice, the unique lowest-degree polynomial satisfying (2.2) can be constructed with the help of the Lagrange fundamental polynomials [19, p. 5]

$$l_{n,i}(x) := \prod_{k=0, k \neq i}^n \frac{x - x_k}{x_i - x_k}, \quad n \geq 1. \quad (2.8)$$

Again, since we assume $x_i \neq x_j \forall i \neq j$, the denominator can not attain zero. Thus, (2.8) is well defined. Note that these functions are polynomials of degree n , with

$$l_{n,i}(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}.$$

Hence, in what follows, it is natural to refer to (2.8) as the *Lagrange basis polynomials*. The *Lagrange polynomial* can now be defined as a linear combination of these basis polynomials:

$$L_n(x) := \sum_{i=0}^n l_{n,i}(x) y_i. \quad (2.9)$$

Consequently, $L_n(x_i) = y_i \forall i = 0, \dots, n$. In other words, we have constructed a polynomial L_n that provides an exact fit to our data points (x_i, y_i) .

As for the matrix formulation (2.3) of the interpolation problem (2.2), inverting the Vandermonde matrix V is not computationally efficient; moreover, as V is often ill-conditioned, the resulting coefficients p_i may be inaccurate [20, p. 283].

The Lagrange basis polynomials (2.8) provide an efficient way of computing the inverse matrix through a change of basis. Specifically, the columns of the Vandermonde matrix are formed by the usual monomial base functions $1, x, x^2, \dots, x^n$ (where $x = [x_0, x_1, x_2, \dots, x_n]^T$). However, if we instead use the basis formed by the Lagrange basis polynomials, the equivalent to the Vandermonde matrix is simply an identity matrix:

$$\begin{bmatrix} l_{n,0}(x_0) & l_{n,1}(x_0) & l_{n,2}(x_0) & \dots & l_{n,n}(x_0) \\ l_{n,0}(x_1) & l_{n,1}(x_1) & l_{n,2}(x_1) & \dots & l_{n,n}(x_1) \\ l_{n,0}(x_2) & l_{n,1}(x_2) & l_{n,2}(x_2) & \dots & l_{n,n}(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n,0}(x_n) & l_{n,1}(x_n) & l_{n,2}(x_n) & \dots & l_{n,n}(x_n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}.$$

Of course, the identity matrix is its own inverse, which makes the inversion step of solving (2.3) trivial. On the other hand, we may want to also obtain the conventional polynomial expression $p(x) = \sum_{i=0}^n p_i x^i$ (i.e., the resulting coefficients in terms of the monomial basis). In that case, some additional calculations are needed, which partially undermines the computational advantage gained through effective matrix inversion.

In either case, there is a more significant caveat to using the Lagrange basis. Namely, every time even one of the nodes x_i changes (or more nodes are added), all Lagrange basis polynomials need to be recomputed, as is evident by looking at their definition (2.8). Nevertheless, this can be mitigated to some degree by employing the barycentric form of the Lagrange polynomial. Specifically, by writing the Lagrange basis polynomials (2.8) as

$$l_{n,i}(x) = w_i \frac{\prod_{k=0, k \neq i}^n (x - x_k)}{x - x_i},$$

where

$$w_i := \frac{1}{\prod_{k=0, k \neq i}^n (x_i - x_k)}$$

are the barycentric weights, the Lagrange polynomial L_n now attains the barycentric form

$$L_n(x) = \prod_{k=0}^n (x - x_k) \sum_{i=0}^n \frac{w_i}{x - x_i} y_i. \quad (2.10)$$

This means that if a node changes, we do not need to recompute all the basis polynomials individually, but only the product $\prod_{k=0}^n (x - x_k)$ and the weights $w_i/(x - x_i)$. Even further computational simplification is possible by leveraging a constant function $h(x) \equiv 1$. Then, by considering (2.10) for both $L_n(x)$ and $h(x)$, we obtain

$$L_n(x) = \frac{L_n(x)}{h(x)} = \frac{\sum_{i=0}^n \frac{w_i}{x-x_i} y_i}{\sum_{i=0}^n \frac{w_i}{x-x_i}}. \quad (2.11)$$

This form, also known as the *true barycentric form*, thus eliminates the product $\prod_{k=0}^n (x - x_k)$ from the calculation. Both barycentric forms (2.10) and (2.11) require only $\mathcal{O}(n)$ operations in order to update a node x_i , as opposed to $\mathcal{O}(n^2)$ operations for the original form (2.9) [21]. Moreover, barycentric interpolation is numerically more stable than the basic Lagrange interpolation [22].

Despite the existence of the barycentric forms of the Lagrange polynomial, it is also worth looking at different ways of constructing the interpolating polynomial.

2.1.3 Non-Lagrangian forms

As shown earlier, the polynomial satisfying (2.2) is unique. While the previous section demonstrated how the problem can be solved in terms of the Lagrange basis, it is not the only possible *form* of the unique solution. Hence, the Lagrange polynomial could more accurately be called the Lagrange form of the polynomial, in contrast with non-Lagrangian forms such as the Newton form and the Bernstein form discussed in this section.

Newton form

The Newton form is based on finite differences, and in fact it is intuitive to think of it as a form of Taylor polynomial, with instantaneous rates of change (i.e., derivatives) replaced with finite differences. As the famous Taylor's theorem states, for a k times differentiable function f at point $a \in \mathbb{R}$, there exists a function h_k such that

$$f(x) = T_k(x) + h_k(x)(x - a)^k \quad \text{and} \quad \lim_{x \rightarrow a} h_k(x) = 0,$$

where $T_k(x)$, the k -th order Taylor polynomial centered at a , equals

$$T_k(x) = f(a) + f'(a)(x - a) + \frac{f^{(2)}(a)}{2}(x - a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x - a)^k. \quad (2.12)$$

Concerning the Newton form itself, we begin by defining the *Newton basis polynomials* as

$$n_i(x) := \begin{cases} 1, & \text{if } i = 0 \\ \prod_{k=0}^{i-1} (x - x_k), & \text{if } i > 0 \end{cases}. \quad (2.13)$$

Then, analogously to the Lagrangian case, the *Newton polynomial* is a linear combination of the Newton basis polynomials (2.13):

$$N(x) := \sum_{i=0}^n n_i(x) y'_i. \quad (2.14)$$

However, the crucial difference between (2.9) and (2.14) is that whereas in the former we simply have the values y_i , in the latter we have the (*forward*) *divided differences*

$$y'_i := D[y_0, y_1, \dots, y_i].$$

This difference is recursively defined as

$$\begin{aligned} D[y_i] &:= y_i, \quad 0 \leq i \leq n \\ D[y_i, \dots, y_j] &:= \frac{D[y_{i+1}, \dots, y_{i+j}] - D[y_i, \dots, y_{i+j-1}]}{x_{i+j} - x_i}, \quad 0 \leq i \leq n-j, \quad 1 \leq j \leq n. \end{aligned}$$

To illustrate this, the explicit forms of $D[y_0, y_1]$ and $D[y_0, y_1, y_2]$ are computed below:

$$\begin{aligned} D[y_0, y_1] &= \frac{D[y_1] - D[y_0]}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}, \\ D[y_0, y_1, y_2] &= \frac{D[y_1, y_2] - D[y_0, y_1]}{x_2 - x_0} = \frac{y_2 - y_1}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_1 - y_0}{(x_2 - x_0)(x_1 - x_0)}. \end{aligned}$$

In general, the Newton tableau [21] visualizes how the computation of $D[y_0, \dots, y_i]$ proceeds term by term, from left to right:

$$\begin{array}{ccccccc} D[y_0] & & & & & & \\ & D[y_0, y_1] & & & & & \\ D[y_1] & & D[y_0, y_1, y_2] & & & & \\ & D[y_1, y_2] & & D[y_0, \dots, y_3] & & & \\ D[y_2] & & D[y_1, y_2, y_3] & & \ddots & & \\ & D[y_2, y_3] & & \vdots & & D[y_0, \dots, y_i] & \\ D[y_3] & & & & \ddots & & \\ & & & \vdots & & D[y_{n-3}, \dots, y_n] & \\ & & & & & \vdots & \\ & & & & & D[y_{n-2}, y_{n-1}, y_n] & \\ & & \vdots & & & & \\ \vdots & D[y_{n-1}, y_n] & & & & & \\ D[y_n] & & & & & & \end{array}$$

By rewriting the Newton polynomial (2.14) as

$$N(x) := y_0 + D[y_0, y_1](x - x_0) + D[y_0, y_1, y_2](x - x_0)(x - x_1) + \dots \\ + D[y_0, \dots, y_n] \prod_{k=0}^{n-1} (x - x_k),$$

its similarity with the Taylor polynomial (2.12) becomes apparent. In particular, in the limit case when all nodes x_0, \dots, x_n approach a single point a , the divided differences turn into derivatives, and we obtain precisely the Taylor polynomial (2.12). Note that as opposed to the (non-barycentric) Lagrange form, a node change does not enforce a recomputation of all the basis functions in the Newton form.

If instead of matching only the values y_i we require also the first m derivatives (up to $y_i^{(m)}$) to match at the given nodes, the method of divided differences is called Hermite interpolation. This is also a natural approach, considering how the Taylor polynomial is constructed.

Bernstein form

Another way of constructing the interpolating polynomial is through a basis formed by the *Bernstein basis polynomials*

$$b_{n,i}(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad i = 0, \dots, n, \quad (2.15)$$

with the binomial coefficient equalling

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}.$$

Explicitly, the basis polynomials of degrees $n = 0, 1, 2, 3$ are

$$\begin{aligned} b_{0,0}(x) &= 1 \\ b_{1,0}(x) &= 1 - x, \quad b_{1,1}(x) = x \\ b_{2,0}(x) &= (1-x)^2, \quad b_{2,1}(x) = 2x(1-x), \quad b_{2,2}(x) = x^2 \\ b_{3,0}(x) &= (1-x)^3, \quad b_{3,1}(x) = 3x(1-x)^2, \quad b_{3,2}(x) = 3x^2(1-x), \quad b_{3,3}(x) = x^3. \end{aligned}$$

Moreover, $b_{n,i} \geq 0$ for $0 \leq x \leq 1$, and for the extrema $x = 0$ and $x = 1$ we have

$$b_{n,i}(0) = \begin{cases} 1, & \text{if } i = 0 \\ 0, & \text{if } i \neq 0 \end{cases}, \quad b_{n,i}(1) = \begin{cases} 1, & \text{if } n = i \\ 0, & \text{if } n \neq i \end{cases}.$$

A linear combination of the basis polynomials (2.15) yields the *Bernstein polynomial*

$$B_n(x) := \sum_{i=0}^n b_{n,i}(x) g\left(\frac{i}{n}\right). \quad (2.16)$$

However, this implies that we should know the target function g , or at least the values $g\left(\frac{i}{n}\right)$, which are also known as the Bernstein coefficients (or Bézier coefficients). Restricting the range to $0 \leq x \leq 1$ creates an important special case of the Bernstein polynomials; in that case, they form the foundation of Bézier curves, to which we will briefly return in the end of Section 2.1.6.

2.1.4 Polynomial interpolation error

In order to quantify the accuracy of polynomial interpolation and to understand its limitations, let us analyze the error associated with this approach. The following theorem is formulated based on the theorem presented in [20, p. 284].

Theorem 1

Let g be an $n+1$ times continuously differentiable function in the interval $[a, b]$ (i.e., $g \in \mathcal{C}^{n+1}[a, b]$), and let p be a polynomial of degree $\leq n$ that interpolates g at $n+1$ distinct nodes x_0, x_1, \dots, x_n in the interval $[a, b]$. Then, for each $t \in [a, b]$, there exists a $\xi_t \in [a, b]$ such that

$$g(t) - p(t) = \frac{\prod_{i=0}^n (t - x_i)}{(n+1)!} g^{(n+1)}(\xi_t). \quad (2.17)$$

Proof

If t is equal to any of the nodes x_i , then (2.17) is trivially true, as $p(x_i) = g(x_i)$ by definition, and thus, both sides of the equation are 0. Assuming $t \neq x_i$ is fixed, let

$$\varepsilon(x) := g(x) - p(x), \quad w(x) := \prod_{i=0}^n (x - x_i)$$

and

$$\phi(x) := \varepsilon(x) - \frac{w(x)}{w(t)} \varepsilon(t)$$

for any value $x \in [a, b]$. Then, since the denominator $w(t)$ does not attain 0 and $\varepsilon \in \mathcal{C}^{n+1}[a, b]$, also $\phi \in \mathcal{C}^{n+1}[a, b]$. Moreover, ϕ has $n+2$ distinct zeros in $[a, b]$, because

$$\phi(t) = \varepsilon(t) - \varepsilon(t) = 0$$

and

$$\phi(x_i) = \varepsilon(x_i) - \frac{\prod_{i=0}^n (x_i - x_i)}{\prod_{i=0}^n (t - x_i)} \varepsilon(t) = \varepsilon(x_i) = 0, \quad i = 0, 1, \dots, n.$$

Consequently, according to Rolle's theorem, ϕ' has $n + 1$ distinct zeros in $[a, b]$. Through induction, $\phi^{(2)}$ has n distinct zeros in $[a, b]$, and so forth, up to $\phi^{(n+1)}$, which has one distinct zero in $[a, b]$. Let ξ_t denote this zero, in other words

$$\phi^{(n+1)}(\xi_t) = 0. \quad (2.18)$$

Since p is a polynomial of degree $\leq n$,

$$\varepsilon^{(n+1)}(x) = g^{(n+1)}(x) - p^{(n+1)}(x) = g^{(n+1)}(x) \quad (2.19)$$

Moreover, leveraging the general Leibniz rule, which in this particular case (computing the $(n + 1)$ -th derivative of the product of $n + 1$ functions) is relatively simple, yields

$$w^{(n+1)}(x) = \frac{(n+1)!}{\underbrace{1!1! \cdots 1!}_{n+1 \text{ terms}}} \frac{d}{dx}(x - x_i) = (n+1)! . \quad (2.20)$$

Using (2.19) and (2.20), we can write

$$\phi^{(n+1)}(x) = g^{(n+1)}(x) - \frac{(n+1)!}{w(t)} \varepsilon(t). \quad (2.21)$$

For $x = \xi_t$, the left-hand side of (2.21) equals 0 by definition (2.18). Hence, by solving (2.21) for $\varepsilon(t)$, we obtain

$$\varepsilon(t) = \frac{w(t)}{(n+1)!} g^{(n+1)}(\xi_t),$$

which equals (2.17). ■

In practice, Theorem 1 implies that the maximum error between g and its polynomial interpolant p equals

$$\max_{x_0 \leq x \leq x_n} |g(x) - p(x)| = \frac{\max_{x_0 \leq x \leq x_n} (\prod_{i=0}^n |x - x_i|)}{(n+1)!} \max_{x_0 \leq \xi_x \leq x_n} |g^{(n+1)}(\xi_x)|. \quad (2.22)$$

For instance, if we wish to approximate $\cos(x)$ by a polynomial of degree $n = 4$ in the interval $[0, 1]$, the maximum approximation error is

$$\max_{0 \leq x \leq 1} \left| \cos(x) - \sum_{i=0}^4 p_i x^i \right| \leq \frac{1}{5!} = \frac{1}{120}$$

due to the upper bounds $\prod_{i=0}^4 |x - x_i| \leq 1$ and $|\cos^{(5)}(\xi_x)| \leq 1$.

Runge's phenomenon

Up to this stage, we have not commented on the spacing of the nodes x_i , besides requiring that each of them is truly a different point, i.e., $x_i \neq x_j \forall i \neq j$. A straightforward option is to assume equidistant spacing between the nodes, but it turns out that such an approach is not optimal in terms of minimizing the maximum approximation error (2.22). In fact, there are pathological cases, for which the error somewhat unintuitively increases as the degree of the interpolating polynomial increases.

This problem is best described through examples. First, let us look at the exponential function e^{-x^2} in the interval $x \in [-2, 2]$, and three of its polynomial interpolants, of degrees $n = 2, 5, 8$. For each degree n , we take $n + 1$ equispaced nodes $x_i \in [-2, 2]$, that is,

$$x_i = \frac{2i - n}{n} \cdot 2, \quad i = 0, 1, \dots, n,$$

and compute the n -th degree polynomial that interpolates the corresponding function values $e^{-x_i^2}$. Figure 2.1 shows the target exponential function and the three interpolants. The second degree polynomial intersects the target function at both endpoints and the middle point, but due to its parabolic shape, the approximation is otherwise poor. The 5th degree polynomial is already a better fit, and it intersects the target function at all six nodes x_0, \dots, x_5 , as it should. Nevertheless, the 8th degree polynomial is clearly the best fit out of the three interpolants. Overall, the largest approximation errors are in the vicinity of the endpoints of the interval, but the accuracy evidently improves as the polynomial degree n increases. However, another example shows that increasing the polynomial degree does not always yield increased approximation accuracy. Consider the famous Runge function

$$g(x) = \frac{1}{1 + x^2}, \quad -5 \leq x \leq 5, \quad (2.23)$$

investigated by Carl Runge in 1901 [20, p. 288] in the very same context of approximating functions via polynomial interpolation. Let us take $n + 1$ equispaced nodes x_i in the interval $[-5, 5]$, in other words

$$x_i = \frac{2i - n}{n} \cdot 5, \quad i = 0, 1, \dots, n,$$

and compute the polynomial interpolants of degrees 5, 7 and 11 of g . The results are shown in Figure 2.2. Even though a higher degree polynomial still yields a better

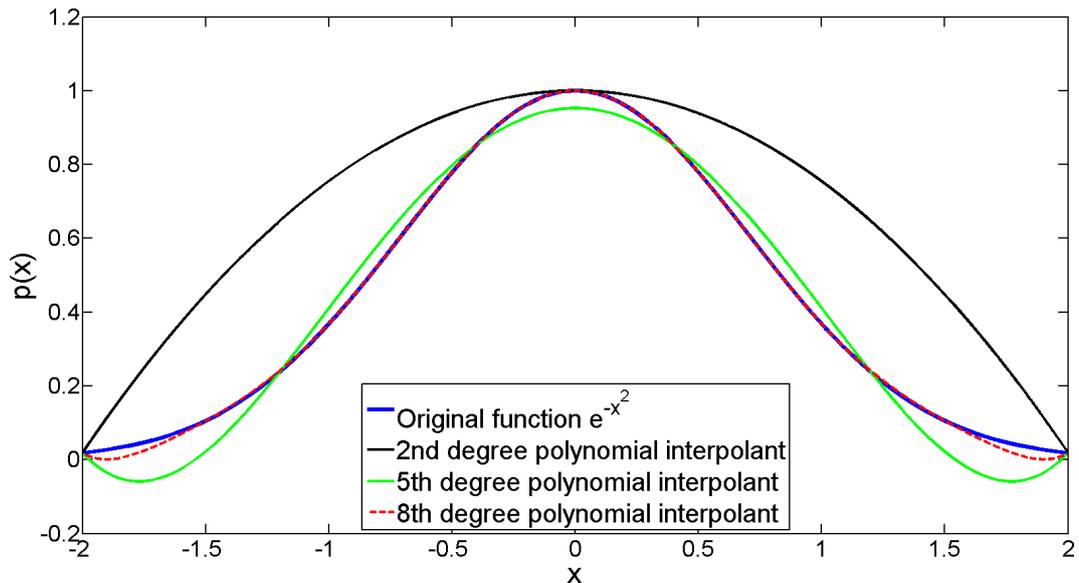


Figure 2.1: Exponential function e^{-x^2} for $x \in [-2, 2]$, and its polynomial interpolants of degrees 2, 5 and 8.

approximation than a lower degree polynomial around the center of the interval, the areas near the endpoints exhibit quite the opposite behaviour: as n increases, the interpolants begin to oscillate more, resulting in a significant increase in the approximation error. Specifically, for any value $3.64 < |x| < 5$ and a sequence of polynomials $p_n(x)$ with increasing degree n , which are constructed as in our example,

$$\sup_{n \geq k} |g(x) - p_n(x)| = \infty, \quad k \geq 0, \quad (2.24)$$

so as $n \rightarrow \infty$, the sequence $p_n(x)$ does not converge to $g(x)$ for any x in the said interval [15, p. 158]. In terms of the maximum error (2.22), the divergence is caused by the fact that the maximum absolute value of the n -th derivative $g^{(n)}(x)$ of the Runge function increases rapidly as n increases.

It is important to clarify that (2.24) only means we can not approximate the Runge function uniformly in the interval $[-5, 5]$ with a polynomial and nodes constructed in this particular way. There is no contradiction with the Weierstrass approximation theorem mentioned in the beginning of Section 2.1, which states that a uniformly converging polynomial does exist, although the theorem does not provide a means to construct it. For an in-depth discussion of more general convergence results, we refer to [20, Ch. 6].

Returning to the problem of node spacing, if $n + 1$ equispaced nodes x_i are used, there is no way to manipulate the product term $\prod_{i=0}^n |x - x_i|$ in the error formula (2.22) to our advantage. In order to reduce the maximum error (2.22), we have two main approaches. The first one is to look for a set of $n + 1$ variably spaced

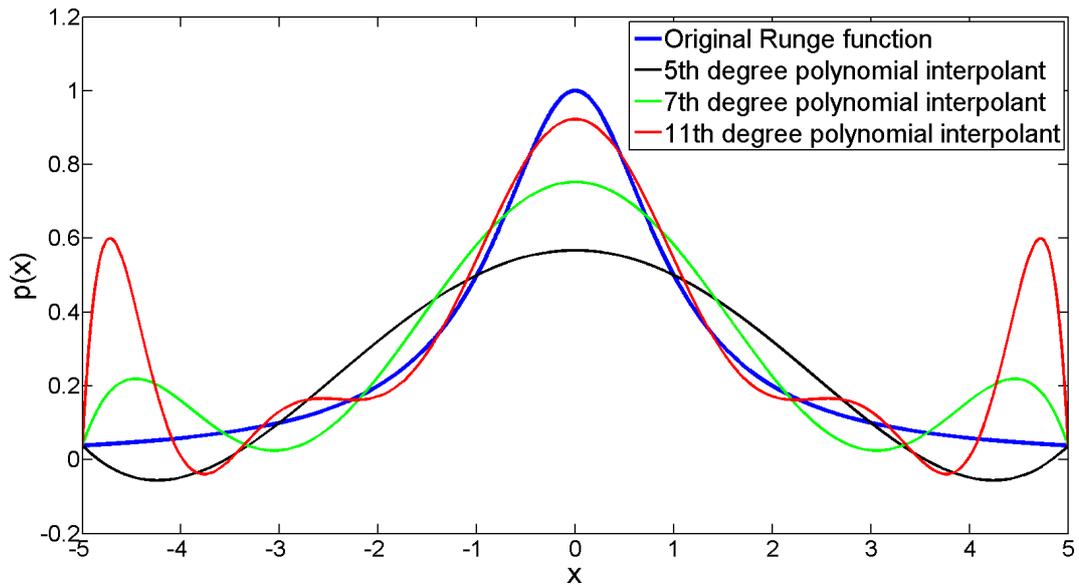


Figure 2.2: Runge function (2.23) for $x \in [-5, 5]$, and its polynomial interpolants of degrees 5, 7 and 11.

nodes x_i that minimize $\prod_{i=0}^n |x - x_i|$. This *Chebyshev node* approach is discussed in Section 2.1.5. The second one is to divide the range into several subintervals and construct a piecewise polynomial fit, with suitable smoothness conditions at the points where the subintervals are joined together. This *spline interpolation* approach is discussed in Section 2.1.6.

2.1.5 Chebyshev nodes

The second order linear differential equations

$$\begin{aligned} (1 - x^2)y^{(2)} - xy' + n^2y &= 0, \\ (1 - x^2)y^{(2)} - 3xy' + n(n + 2)y &= 0, \end{aligned} \quad (2.25)$$

are commonly known as the Chebyshev differential equations. In particular, their solutions are known as the *Chebyshev polynomials* of the first kind and second kind, respectively. We are interested only in the first kind, so in what follows, they are simply referred to as Chebyshev polynomials. We denote an n -th degree Chebyshev polynomial by $T_n(x)$, and they are recursively defined as

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \quad n \geq 2. \end{aligned} \quad (2.26)$$

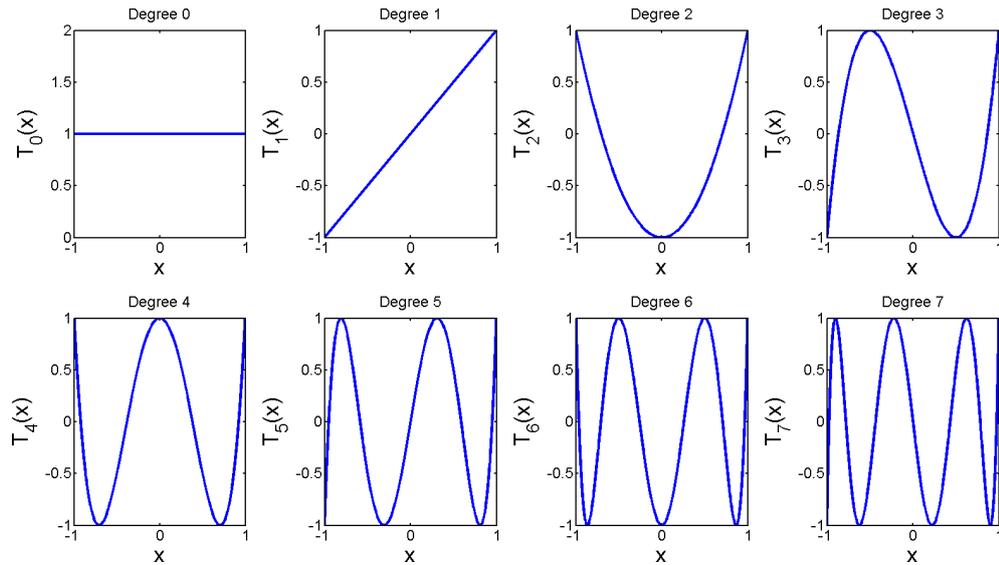


Figure 2.3: Chebyshev polynomials $T_n(x)$ of degrees $n = 1, \dots, 7$ in the interval $x \in [-1, 1]$.

The recursive definition yields

$$\begin{aligned} T_2(x) &= 2x^2 - 1, & T_5(x) &= 16x^5 - 20x^3 + 5x, \\ T_3(x) &= 4x^3 - 3x, & T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1, \\ T_4(x) &= 8x^4 - 8x^2 + 1, & T_7(x) &= 64x^7 - 112x^5 + 56x^3 - 7x, \end{aligned}$$

for the Chebyshev polynomials up to degree 7, to give a better idea of their explicit form. While not immediately obvious from the expressions above, these polynomials are particularly interesting in the range $x \in [-1, 1]$. From Figure 2.3, which visualizes the polynomials $T_0(x), \dots, T_7(x)$ restricted to the said interval, we see the emergence of sinusoidal patterns. Specifically, when $x \in [-1, 1]$, the Chebyshev polynomials can be written in the non-recursive form

$$T_n(x) = \cos(n \cdot \arccos x), \quad n \geq 0, \quad (2.27)$$

which can be proven by using the cosine addition formula $\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$ [20, p. 285].

From (2.27) it is clear that $-1 \leq T_n(x) \leq 1$ for $x \in [-1, 1]$. Let us further denote $\varphi := \arccos(x)$, which means $T_n(x) = \cos(n\varphi)$ for $\varphi \in [0, \pi]$. Then, T_n attains its extreme values ± 1 at $\cos(n\varphi) = \pm 1$, in other words when

$$\varphi = \frac{i\pi}{n}, \quad i = 0, \dots, n.$$

Substituting these values of φ into (2.27) yields

$$T_n(x) = \cos\left(n \cdot \frac{i\pi}{n}\right) = (-1)^i, \quad i = 0, \dots, n,$$

showing that T_n attains these $n + 1$ extreme values -1 and 1 in an alternating fashion. Hence, according to the intermediate value theorem, $T_n(x)$ has n distinct roots in $x \in [-1, 1]$. These roots are located at points $\cos(n\varphi) = 0$, or equally

$$\varphi = (2i + 1)\frac{\pi}{2n}, \quad i = 0, \dots, n - 1.$$

As per our notation, we are interested in $n + 1$ interpolation nodes. Thus, we finally define the *Chebyshev nodes* x_0, \dots, x_n as the roots of the Chebyshev polynomial $T_{n+1}(x)$ in the interval $x \in [-1, 1]$:

$$x_i = \cos\left(\frac{2i + 1}{2(n + 1)}\pi\right), \quad i = 0, \dots, n. \quad (2.28)$$

Note that the restriction to the interval $[-1, 1]$ does not preclude more general usage of the nodes, as they can be computed for an arbitrary interval $[a, b]$ with a simple affine transformation

$$x_i = \frac{1}{2}(a + b) + \frac{1}{2}(b - a) \cos\left(\frac{2i + 1}{2(n + 1)}\pi\right), \quad i = 0, \dots, n. \quad (2.29)$$

Now we can return to the interpolation error term $w(x) = \prod_{i=0}^n (x - x_i)$ discussed in the previous section. Specifically, for any monic polynomial $p(x)$ of degree n (i.e., a polynomial whose highest-degree coefficient is $p_n = 1$), the inequality

$$\|p\|_\infty = \max_{-1 \leq x \leq 1} |p(x)| \geq 2^{1-n}$$

applies [20, p. 286]. Since $w(x)$ is a monic polynomial of degree $n + 1$, we have a lower bound

$$\max_{-1 \leq x \leq 1} \left| \prod_{i=0}^n (x - x_i) \right| \geq 2^{-n} \quad (2.30)$$

for the maximum error associated with the interpolation nodes x_i . When these x_i are chosen to be the Chebyshev nodes (2.28), in other words the roots of $T_{n+1}(x)$, we have

$$w(x) = \prod_{i=0}^n (x - x_i) = \frac{T_{n+1}(x)}{p_{n+1}},$$

where p_{n+1} is the highest-degree coefficient of $T_{n+1}(x)$. Using the recursive definition (2.26) of $T_{n+1}(x)$, it is easy to inductively prove that $p_{n+1} = 2^n$ for any $n \geq 0$.

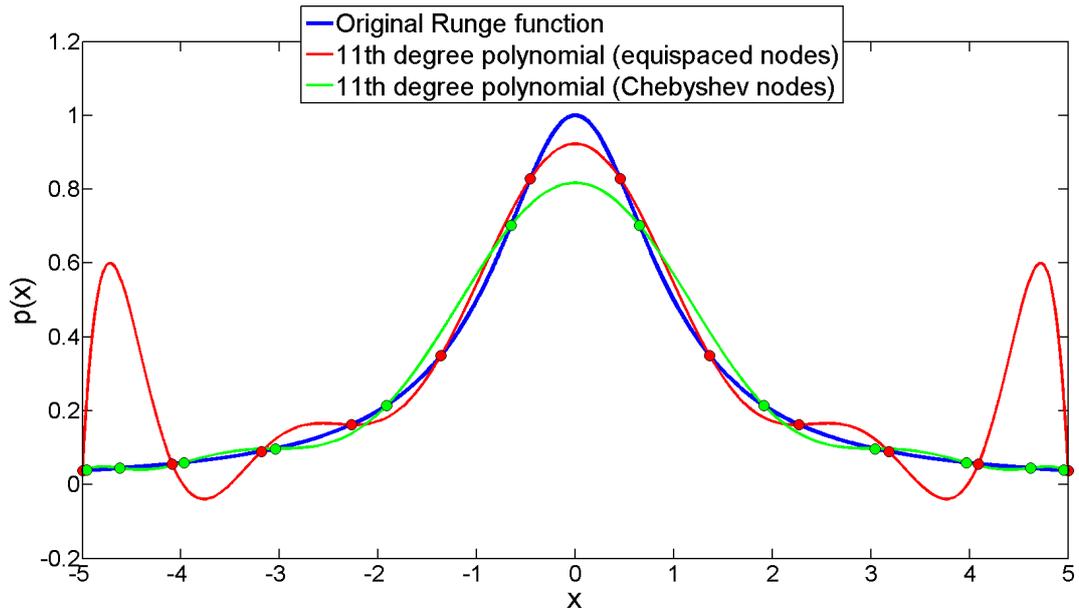


Figure 2.4: Runge function (2.23) for $x \in [-5, 5]$, and its polynomial interpolant of degree 11, with equispaced nodes (red) and Chebyshev nodes (green).

In other words, $2^{-n}T_{n+1}(x)$ is a monic polynomial. It follows that

$$\max_{-1 \leq x \leq 1} \left| \prod_{i=0}^n (x - x_i) \right| = \max_{-1 \leq x \leq 1} |2^{-n}T_{n+1}(x)| = 2^{-n} \cdot \max_{-1 \leq x \leq 1} |T_{n+1}(x)| = 2^{-n},$$

which proves that the lower bound 2^{-n} in (2.30) is reached by using the Chebyshev nodes x_i . In summary, the Chebyshev nodes provide the optimal spacing for the interpolation nodes, with regard to the maximum error (2.22).

As Figure 2.2 demonstrated, equispaced nodes result in a poor interpolation performance regarding the Runge function, especially when the polynomial degree n increases. Figure 2.4 illustrates, for the case $n = 11$, how the interpolant improves when Chebyshev nodes are used instead. In particular, the Chebyshev nodes (marked with green circles) are more closely spaced near the edges, which helps in dramatically dampening the oscillations associated with the equispaced nodes (red circles), even though it comes at the cost of a slightly less accurate approximation in the middle of the interval.

2.1.6 Spline interpolation

Even though the Chebyshev nodes provide an improvement over equidistant nodes, both methods still have the restriction of having only one polynomial to be optimized over the whole range. Hence, this section explores the idea of splitting the range into smaller subintervals and fitting a polynomial separately for each of them, and creating a piecewise polynomial interpolant with smooth transitions between the

subintervals, or in short, a *spline*. In fact, this idea stems from flexible mechanical rulers used by architects and designers to draw smooth curves [23, p. 235]. We concentrate on two common types of splines by providing a detailed description of cubic splines and briefly introducing B-splines. For much more information on splines, we refer the reader to the authoritative book [24] by Carl de Boor.

A general definition of a spline goes as follows. Take $n + 1$ distinct points $x_0 < x_1 < \dots < x_n$, which are called *knots*, and an integer m . Then, a function $S^m : [x_0, x_n] \rightarrow \mathbb{R}$ is a spline of degree m (or order m), if it satisfies the following two conditions:

- (I) S^m is a polynomial of degree $\leq m$ in each half-open subinterval $[x_{i-1}, x_i)$, $i = 1, \dots, n$,
- (II) S^m is $m - 1$ times continuously differentiable in the whole interval $[x_0, x_n]$, i.e., $S^m \in \mathcal{C}^{m-1}[x_0, x_n]$.

In other words, a spline S^m is a piecewise continuous $m - 1$ times differentiable polynomial of degree m . As the most simple examples, splines of degree 0 are piecewise constant functions, and splines of degree 1 are piecewise affine continuous functions. Note that neither of these splines are generally differentiable at the knots, as condition (II) only concerns the differentiability of splines of degree 2 and higher. However, that is also a reason why these lowest-degree cases are often not very useful, as they do not guarantee sufficient smoothness of the spline; the cubic splines provide more flexibility.

Cubic splines

Let us have a closer look at solving the now familiar interpolation problem by using a cubic spline S^3 . Our treatment follows loosely the analysis presented in [20, Ch. 6]. For notational simplicity, we ignore the superscript in what follows, and assume that $S := S^3$. We have the $n + 1$ target values y_0, \dots, y_n , and we wish to construct a cubic spline S that interpolates these values and has $n + 1$ knots $x_0 < x_1 < \dots < x_n$. Hence, we need to construct a sequence of n polynomials S_0, \dots, S_{n-1} such that

$$S(x) = \begin{cases} S_0(x), & x \in [x_0, x_1] \\ S_1(x), & x \in [x_1, x_2] \\ \vdots & \vdots \\ S_{n-1}(x), & x \in [x_{n-1}, x_n] \end{cases}. \quad (2.31)$$

Note that we may use closed intervals $[x_{i-1}, x_i]$ instead of half-open ones $[x_{i-1}, x_i)$, since the interpolation conditions

$$S_{i-1}(x_i) = S_i(x_i) = y_i, \quad i = 1, \dots, n-1$$

enforce continuity at each interior knot. For each of the n subintervals $[x_{i-1}, x_i]$, $i = 1, \dots, n$, we have also $S_{i-1}(x_{i-1}) = y_{i-1}$ and $S_{i-1}(x_i) = y_i$. In addition, the definition of spline S requires that the first and second derivatives (S' and $S^{(2)}$, respectively) are also continuous. Summarizing the conditions, we have

$$S_{i-1}(x_{i-1}) = y_{i-1}, \quad i = 1, \dots, n, \quad (2.32)$$

$$S_{i-1}(x_i) = y_i, \quad i = 1, \dots, n, \quad (2.33)$$

$$S'_{i-1}(x_i) = S'_i(x_i), \quad i = 1, \dots, n-1, \quad (2.34)$$

$$S^{(2)}_{i-1}(x_i) = S^{(2)}_i(x_i), \quad i = 1, \dots, n-1. \quad (2.35)$$

Each polynomial S_i (of degree 3 at most) has four coefficients, so the spline consists of $4n$ coefficients in total. On the other hand, equations (2.32)–(2.35) provide $4n-2$ conditions, so our solution is not strictly determined by the conditions, but we are left with 2 degrees of freedom.

Next, we will construct the polynomial $S_i(x)$, $x \in [x_i, x_{i+1}]$. Let $S^{(2)}(x_i) := d_i$ and $\Delta_i := x_{i+1} - x_i$. Since $S^{(2)}$ is continuous at each interior knot, d_i is the limit

$$\lim_{x \rightarrow x_i} S^{(2)}(x) = d_i, \quad i = 1, \dots, n-1. \quad (2.36)$$

On the other hand, as S_i is cubic, $S_i^{(2)}$ is linear, with the subinterval endpoints satisfying $S_i^{(2)}(x_i) = d_i$ and $S_i^{(2)}(x_{i+1}) = d_{i+1}$. Then, writing out the equation for the line between points (x_i, d_i) and (x_{i+1}, d_{i+1}) yields

$$S_i^{(2)}(x) = \frac{d_i}{\Delta_i}(x_{i+1} - x) + \frac{d_{i+1}}{\Delta_i}(x - x_i).$$

By integrating twice with respect to x , we obtain

$$S'_i(x) = -\frac{d_i}{2\Delta_i}(x_{i+1} - x)^2 + \frac{d_{i+1}}{2\Delta_i}(x - x_i)^2 + (C_1 - C_2), \quad (2.37)$$

$$S_i(x) = \frac{d_i}{6\Delta_i}(x_{i+1} - x)^3 + \frac{d_{i+1}}{6\Delta_i}(x - x_i)^3 + C_1(x - x_i) + C_2(x_{i+1} - x). \quad (2.38)$$

Note that for convenience, the constants of integration ($C_1 - C_2$ in (2.37) and $C_2x_{i+1} - C_1x_i$ in (2.38)) have been chosen so that the last two terms of $S_i(x)$ resemble its first two terms. Now C_1 and C_2 can be easily determined by respectively

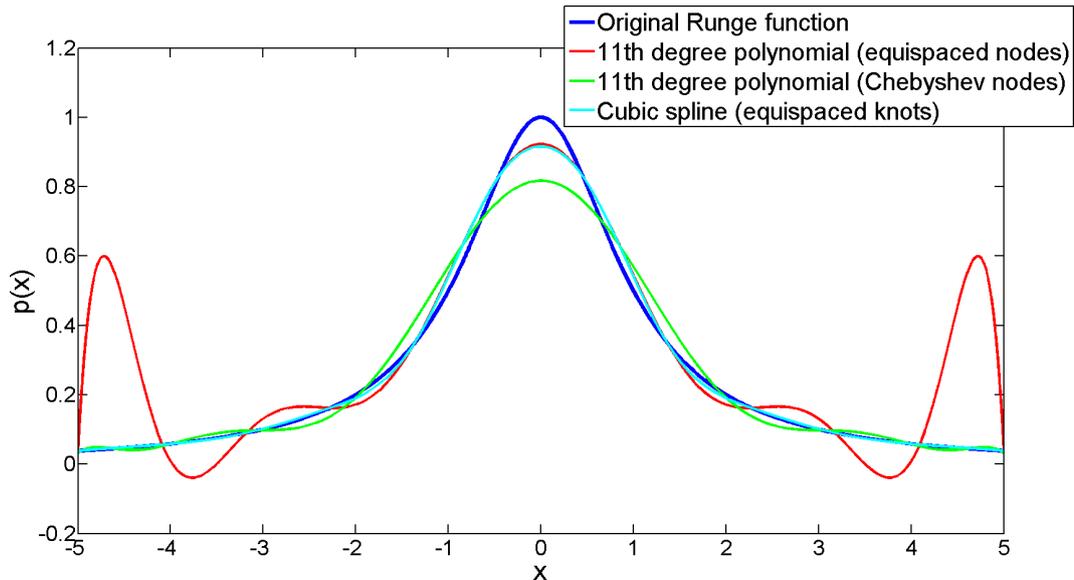


Figure 2.5: Runge function (2.23) for $x \in [-5, 5]$, its polynomial interpolant of degree 11 with either equispaced nodes or Chebyshev nodes, and the cubic spline interpolant.

There are efficient algorithms for computing the derivatives d_i via this system, for instance a Gaussian elimination algorithm without scaled row pivoting [20, p. 319].

To conclude the analysis, we mention an interesting result concerning these natural cubic splines (i.e., splines with $d_0 = d_n = 0$). Specifically, they are the smoothest possible interpolating functions, in the sense that they minimize the (approximate) curvature

$$\int_{x_0}^{x_n} [S^{(2)}(x)]^2 dx$$

over all continuous functions in the interval $[x_0, x_n]$ [20, p. 319].

Finally, let us once more look at the example of interpolating the Runge function, in order to see how the cubic spline performs. For solving the spline coefficients, we use the `spline` function in Matlab; it also constructs and solves a tridiagonal linear system (which is not necessarily restricted by the natural conditions $d_0 = d_n = 0$), as presented in [24].

A comparison of the results is shown in Figure 2.5. The knots for the spline are the same as the 12 equidistant nodes for the polynomial. Evidently, the cubic spline produces the best interpolation result: it dampens the oscillations near the edges even better than the polynomial interpolant with Chebyshev nodes, and it does this without compromising its performance in the middle of the interval. We did not enforce the conditions $d_0 = d_n = 0$ on the pictured spline, although in this example there appeared to be no significant advantage in using or not using such conditions.

B-splines

B-splines are a way of representing splines in terms of basis splines B_i , hence their name. Specifically, assuming a set of infinitely many knots

$$\dots < x_{-2} < x_{-1} < x_0 < x_1 < x_2 < \dots,$$

the B-splines of degree 0 are defined as

$$B_i^0(x) = \begin{cases} 1, & x_i \leq x < x_{i+1} \\ 0, & \text{otherwise} \end{cases}. \quad (2.43)$$

We see from the definition that B_i^0 is continuous from the right, always non-negative, and its support (i.e., the set $\{x : B_i^0(x) \neq 0\}$) is the half-open interval $[x_i, x_{i+1})$. In addition, $\sum_{i=-\infty}^{\infty} B_i^0(x) = 1$ for all x , as each x can occupy only one of the intervals. Given a set of specific knots, the B-splines (2.43) provide a basis for all (right-continuous) splines S of degree 0 formed by using those knots. In particular, every such S is a piecewise constant function with

$$S(x) = c_i, \quad x_i \leq x < x_{i+1}$$

for all (infinitely many) integers i . Hence, the infinite series

$$S(x) = \sum_{i=-\infty}^{\infty} c_i B_i^0(x) \quad (2.44)$$

shows that the basis splines B_i^0 do indeed form a base, in the sense that each element of the space has a unique representation as an infinite series [20, p. 334].

All the higher-order B-splines are defined recursively, starting from the B-splines of degree 0. Let us introduce auxiliary functions

$$V_i^n(x) = \frac{x - x_i}{x_{i+n} - x_i}.$$

Then, the B-splines of degree n are defined as

$$B_i^n(x) = V_i^n B_i^{n-1} + (1 - V_{i+i}^n) B_{i+1}^{n-1}. \quad (2.45)$$

Due to the linearity of V_i^n , the definition (2.45) implies that the piecewise polynomial B_i^n is of degree $\leq n$. As for some of the general properties of the B-splines, we have that [20, pp. 335–337],

$$(I) \quad B_i^n(x) = 0, \quad \text{if } x \notin (x_i, x_{i+n+1}), \quad k \geq 1,$$

$$(II) B_i^n(x) > 0, \text{ if } x \in (x_i, x_{i+n+1}), k \geq 0,$$

$$(III) \sum_{i=-\infty}^{\infty} B_i^n(x) = 1,$$

$$(IV) B_i^n \in \mathcal{C}^{n-1}(\mathbb{R}),$$

$$(V) S(x) = \sum_{i=-\infty}^{\infty} C_i B_i^0(x) \text{ for given coefficients } C_i^n,$$

generalizing the earlier result (2.44) of representing any spline as an infinite series using the B-splines.

Regarding the approximation properties of B-splines, the result known as Marsden's identity shows that any polynomial can be represented as a linear combination of B-splines, given an infinite knot sequence [25, p. 59]. Further, Schoenberg's scheme is an effective method of approximating (not interpolating) smooth functions with B-splines [25, p. 77].

Another interesting observation is that B-splines are closely related to the Bernstein polynomials (2.16). In particular, a B-spline B_i^n with the knots

$$0 = x_i = \dots = x_n < x_{n+1} = \dots = x_{n+i+1} = 1$$

is exactly the Bernstein basis polynomial (i.e., a basis function of a Bézier curve)

$$b_{n,i}(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad 0 \leq x < 1,$$

which can be proven inductively [25, p. 54].

Bézier curves, B-splines, and their generalizations, non-uniform rational B-splines (NURBS) are widely used for example in computer aided design and computer graphics [26; 27; 28], and to mention a specific example, in font design [29, p. 203].

2.2 Rational function interpolation

In the previous section, we discussed various approaches to polynomial interpolation. However, there are situations where polynomial interpolation is unlikely to produce good approximations. These include scenarios where the target function has a singularity and thus becomes infinite at some finite value $x = a$ (very rapid growth, e.g., $g(x) = x^{-1}$), the function approaches a finite limit value asymptotically (very slow growth), or the function or its derivatives become infinite or discontinuous at any point [23, p. 217]. Hence, here we expand our point of view into rational functions, that is, ratios of polynomials.

The field of rational interpolation and rational approximation is somewhat more recent than the extensively studied topic of polynomial interpolation. Nevertheless, Cauchy explored rational interpolation as early as in 1821 [23, p. 223], and

Chebyshev began to investigate rational approximation in his 1859 paper on function approximation [30]. Chebyshev discussed a generic minimax formulation of the problem (i.e., minimizing the maximum value of a function), but soon narrowed his focus to polynomial, weighted and rational approximation [31, p. 38]. For more history of approximation theory, we refer to [31].

A rational function $R(x)$ is a ratio of two polynomials $P(x)$ and $Q(x)$:

$$R(x) := \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^N p_i x^i}{\sum_{i=0}^M q_i x^i}, \quad (2.46)$$

where the degrees of $P(x)$ and $Q(x)$ are N and M , respectively, and the denominator $Q(x)$ is not a zero polynomial ($Q \not\equiv 0$). If the degrees need to be explicitly specified, we denote

$$R_{N,M}(x) := \frac{P_N(x)}{Q_M(x)}.$$

Note that rational functions are a superset of polynomials, as all polynomials can be obtained with (2.46) by setting $Q(x) \equiv 1$. We also assume that any zero of $Q(x)$ is a pole of $R(x)$, meaning that if $Q(x) = 0$ somewhere in the interval $[a, b]$, then $R(x)$ becomes unbounded in that interval [23, p. 219].

Let us examine the interpolation problem for rational functions. Nominally, $R(x)$ has $N + M + 2$ unknown coefficients. However, the coefficients are often scaled such that one of them becomes unitary (say, $q_0 = 1$), which in practice leaves us with $N + M + 1$ independent unknowns. Thus, assume we have $N + M + 1$ data points (x_i, y_i) , $i = 0, \dots, N + M$. Then, we wish to find $R(x)$ such that

$$R(x_i) = y_i, \quad i = 0, \dots, N + M. \quad (2.47)$$

A subtle reformulation of the above yields a system of homogeneous linear equations

$$P(x_i) - y_i Q(x_i) = 0, \quad i = 0, \dots, N + M, \quad (2.48)$$

or explicitly,

$$p_0 + p_1 x_i + \dots + p_N x_i^N - y_i (q_0 + q_1 x_i + \dots + q_M x_i^M) = 0, \quad i = 0, \dots, N + M.$$

It is clear that a solution to (2.47) is also a solution to (2.48). However, an inconvenience in dealing with rational interpolation, as opposed to polynomial interpolation, is that the opposite does not necessarily hold. We shall demonstrate this via a simple example from [32, pp. 58–59].

Take nodes $x_i = 0, 1, 2$ and corresponding values $y_i = 1, 2, 2$, and $N = M = 1$, so

that

$$R(x) = \frac{p_0 + p_1x}{q_0 + q_1x}.$$

Then, the linear equations (2.48) are

$$\begin{aligned} p_0 - q_0 &= 0, \\ p_0 + p_1 - 2(q_0 + q_1) &= 0, \\ p_0 + 2p_1 - 2(q_0 + 2q_1) &= 0. \end{aligned}$$

This gives us $p_0 = q_0 = 0$, $p_1 = 2$ and $q_1 = 1$ (with q_1 normalized to unity). Hence,

$$R(x) = \frac{2x}{x},$$

which is not determined for one of our nodes $x_0 = 0$. In other words, the solution to (2.48) does not satisfy (2.47). In particular, this situation arises because the denominator has a root $Q(x_0) = 0$ at one of the nodes, and $P(x)$ and $Q(x)$ have a common factor x .

In general, since $P(x)$ and $Q(x)$ may have common factors (such as x in the above example), there are many representations of the same rational function. However, it can be proven that all nontrivial solutions of the linear formulation (2.48) determine the same rational function, in the sense that $P(x)Q^*(x) \equiv P^*(x)Q(x)$ for two such solutions $R(x) = P(x)/Q(x)$ and $R^*(x) = P^*(x)/Q^*(x)$ [32, p. 60]. Moreover, if $P(x)$ and $Q(x)$ do not have any common factors, then the (unique) solution of (2.48) is also the unique solution of the original interpolation problem (2.47) [32, p. 61]. In other words, (2.47) either has a unique solution, or does not have any solution. Fortunately, the solvability is characterizable: assuming $n + 1$ node points and the polynomial degrees N and M , problem (2.47) is solvable if the sum of the degrees is low enough, specifically $N + M < n$ (see [32, p. 62] and the references therein). For instance, in the above example we had $N + M = 2 = n$, and there was a common factor between $P(x)$ and $Q(x)$, so there was no solution.

There are various algorithms, mostly iterative in nature, for efficiently computing the solution to the rational interpolation problem; many of the algorithms are reviewed in [33]. Some of them are designed for evaluating the rational polynomial at certain values, and others for solving the coefficients of the rational polynomial itself. Note that even though the naïve approach of solving the homogeneous equations (2.48) failed in the above example, in practice it is still rare to have $Q(x_i) = 0$ [23, p. 224]. Hence, this approach is often enough to produce a solution that is also valid for the original problem, if efficiency is not of concern. As for the more sophisticated approaches, we briefly introduce one presented in [32]. Its concept is similar to the Newton divided differences (2.15) described in Section 2.1.3.

First, we recursively define *inverse differences*

$$D[x_i, x_j] := \frac{x_i - x_j}{y_i - y_j},$$

$$D[x_i, x_j, x_k] := \frac{x_j - x_k}{D[x_i, x_j] - D[x_i, x_k]}, \quad (2.49)$$

$$\vdots$$

$$(2.50)$$

$$D[x_i, \dots, x_l, x_m, x_n] := \frac{x_m - x_n}{D[x_i, \dots, x_m] - D[x_i, \dots, x_n]}.$$

The idea of the method is to construct a tableau of inverse differences as follows [32, p. 63]:

i	x_i	y_i	$D[x_0, x_i]$	$D[x_0, x_1, x_i]$	$D[x_0, x_1, x_2, x_i]$
0	x_0	y_0			
1	x_1	y_1	$D[x_0, x_1]$		
2	x_2	y_2	$D[x_0, x_2]$	$D[x_0, x_1, x_2]$	
3	x_3	y_3	$D[x_0, x_3]$	$D[x_0, x_1, x_3]$	$D[x_0, x_1, x_2, x_3]$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

Let $n := \max\{N, M\}$. Then, for $i = 0, 1, \dots, 2n$, we can derive a continued fraction [32, pp. 64–65]

$$\begin{aligned} R_{n,n}(x) &= x_0 + \frac{x - x_0}{Q_n(x)/P_{n-1}(x)} \\ &= x_0 + \frac{x - x_0}{D[x_0, x_1] + \frac{x - x_1}{P_{n-1}(x)/Q_{n-1}(x)}} \\ &= \dots \\ &= x_0 + \frac{x - x_0}{D[x_0, x_1] + \frac{x - x_1}{D[x_0, x_1, x_2] + \frac{x - x_2}{D[x_0, x_1, x_2, x_3] + \frac{\dots}{\dots + \frac{x - x_{2n-1}}{D[x_0, \dots, x_{2n}]}}}}}. \end{aligned}$$

Finally, truncating this expression yields the rational expressions for lower degrees:

$$\begin{aligned} R_{0,0}(x) &= x_0, \\ R_{1,0}(x) &= x_0 + \frac{x - x_0}{D[x_0, x_1]} \\ R_{1,1}(x) &= x_0 + \frac{x - x_0}{D[x_0, x_1] + \frac{x - x_1}{D[x_0, x_1, x_2]}}, \end{aligned}$$

continuing similarly for $R_{2,1}(x)$, $R_{2,2}(x)$, $R_{3,2}(x)$, and so forth. Interestingly, the same concept of continued fractions can be used for obtaining good rational approximations of real numbers [23, p. 227].

2.3 Function approximation

As commented in Section 2.1, in practice we often have noisy or otherwise inaccurate data $g(x_i) = y_i$. Then, requiring the approximating function f to coincide with a set of exact values may lead to a very high-degree model that is not a good representation of the reality. In other words, we wish for the polynomial, spline or rational function to model the actual physical phenomena underlying the data, instead of overfitting it to adhere to a specific set of measurements containing random noise. Hence, more generally, we are interested in minimizing the approximation error

$$\varepsilon := \|f - g\|_d$$

with respect to metric d , while possibly imposing additional constraints on the approximant f . For instance, we may want a polynomial (or similarly, the numerator and denominator of a rational function) to be of a certain degree. On the other hand, if x is a random variable, we may want the distribution of $f(x)$ to have certain properties, or in our case, the variance of $f(x)$ to be constant.

In Sections 2.3.1–2.3.2, we explore a few options of quantifying the approximation error. Specifically, we consider minimax approximation and (weighted) least squares approximation. For a more general discussion on best approximations in linear spaces, we refer to [20, Ch. 6] and [34, Ch. 3]. However, note that rational functions do not form a linear space [35, p. 56].

Finally, in Section 2.3.3, we present some further thoughts on rational approximation, and in particular contemplate the advantages and disadvantages of rational approximation over polynomial approximation.

2.3.1 Minimax approximation

Recall that according to the Weierstrass approximation theorem, any continuous function can be uniformly approximated by a polynomial in a closed interval. In fact, in the interval $[0, 1]$, the Bernstein polynomials can be used to achieve this, although the convergence is slow [15, p. 198]. On the other hand, if the target function has a Taylor series, it can be truncated in order to obtain a polynomial approximation of degree n . Taylor approximation is often used as a preliminary step for a more accurate approximation method. However, in itself, it is not especially efficient, and the error is typically not distributed evenly along the interval [15, p. 199].

In order to examine the accuracy of a polynomial approximant p in the interval $[a, b]$, we define the minimax error as

$$\varepsilon(g) := \inf_{\deg(p) \leq n} \|p - g\|_{\infty}, \quad (2.51)$$

where

$$\|h\|_{\infty} := \sup_{a \leq x \leq b} |h(x)|$$

is the uniform norm, also known as the maximum norm or the infinity norm. For continuous functions, the infimum and supremum can be replaced with minimum and maximum, respectively. The Chebyshev equioscillation theorem states that for $g \in \mathcal{C}[a, b]$, there exists an optimal unique polynomial p_n^* , for which $\|p_n^* - g\|$ reaches the minimum $\varepsilon(g)$ [15, p. 224]. The Remez algorithm may be employed in order to iteratively construct such an approximation [20, pp. 381–383]; alternatively, one may use near-minimax approximation methods that often yield a good estimate of the optimal polynomial [15, p. 225].

2.3.2 Least squares approximation

As an easier alternative to the minimax approximation methods, we consider the least squares approximation. Analogously to (2.51), the least squares error is defined as

$$\varepsilon_{LS}(g) := \inf_{\deg(p) \leq n} \|p - g\|_2,$$

where

$$\|h\|_2 := \left(\int_a^b |h(x)|^2 dx \right)^{\frac{1}{2}}, \quad h \in \mathcal{C}[a, b]$$

is the L^2 norm. Naturally, if we only have a discrete set of data points $g(x_i) = y_i$, $i = 0, \dots, n$, the expression to be minimized then becomes

$$\sum_{i=0}^n |p(x_i) - y_i|^2.$$

In order to have a more general approach, we may want to favour certain subintervals of the range $[a, b]$ over others. For instance, some of the obtained data may be more reliable than the rest, or obtaining a highly accurate approximation may be more crucial in certain regions. Thus, we define the weighted L^2 norm

$$\|h\|_{2,w} := \left(\int_a^b w(x) |h(x)|^2 dx \right)^{\frac{1}{2}}, \quad h \in \mathcal{C}[a, b],$$

where $w(x) \geq 0$ is a non-negative weight function. In the discrete case, we then have

$$\sum_{i=0}^n w(x_i) |p(x_i) - y_i|^2$$

as the minimization criterion.

Again, an optimal unique polynomial p_n^* minimizing $\|p - g\|_{2,w}$ does exist, and it can be written in the form [15, p. 217]

$$p_n^*(x) = \sum_{i=0}^n \langle g, \phi_i \rangle \phi_i(x),$$

where

$$\langle g, \phi_i \rangle := \int_a^b w(x) g(x) \phi_i(x) dx$$

is the weighted inner product of continuous functions g and ϕ_i , and $\{\phi_i \mid i \geq 0\}$ is an orthonormal basis, implying

$$\langle \phi_i, \phi_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}.$$

In practice, using the simple monomial basis $\{1, x, x^2, \dots\}$ would characteristically lead to an unstable solution [23, pp. 174–175]. Thus, orthonormal bases are preferred, for instance bases formed with either Legendre polynomials or the Chebyshev polynomials T_n [15, pp. 207–222].

2.3.3 Rational approximation

A classical way of approximating a continuous function g with a rational function $R_{N,M}$ is Padé approximation. In particular, it minimizes neither the uniform norm nor the L^2 norm, but instead requires $R_{N,M}(x)$ to agree at some point $x = a$ with the function $g(x)$ and as many of its derivatives ($N + M$ at most) as possible. In other words, we require

$$R(a) = g(a), \quad R'(a) = g'(a), \quad \dots, \quad R^{(N+M)}(a) = g^{(N+M)}(a). \quad (2.52)$$

In the special case of the denominator being $Q_M(x) \equiv 1$ (and thus $R_{N,M}(x)$ being a polynomial), the solution is easily found by setting $R_{N,M}(x)$ to be equal to the first $N + M + 1$ terms of the Taylor series of $g(x)$ expanded at $x = a$ [23, p. 220].

Assuming $g(x)$ has a Taylor series $g(x) = \sum_{i=0}^{\infty} c_n x^n$ at $x = 0$, the requirements (2.52) imply that we want the numerator of the error $g(x) - R_{N,M}(x)$, that is,

$$(q_0 + q_1 x + \dots + q_M x^M)(c_0 + c_1 x + \dots) - (p_0 + p_1 x + \dots + p_N x^N) \quad (2.53)$$

to have a zero of as high multiplicity as possible at $x = 0$. We illustrate this through an example of approximating the exponential function e^x , as in [23, pp. 220–221]. Let

$$R_{1,1}(x) = \frac{p_0 + p_1 x}{q_0 + x},$$

meaning q_1 has been normalized to unity. Now, we want to zero the first $N + M + 1 = 3$ terms of $e^x - R_{1,1}(x)$ at $x = 0$. Since the Taylor series of e^x at $x = 0$ equals

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots,$$

we can write the numerator of $e^x - R_{1,1}(x)$ as

$$\begin{aligned} e^x(q_0 + x) - (p_0 + p_1 x) &= (1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots)(q_0 + x) - p_0 - p_1 x \\ &= (q_0 - p_0) + (1 - q_0 - p_1)x + (\frac{q_0}{2} + 1)x^2 + \\ &\quad + (\frac{q_0}{6} + \frac{1}{2})x^3 + \dots \end{aligned}$$

Zeroing the first three coefficients and solving the resulting linear equations yields $p_0 = -2$, $p_1 = -1$ and $q_0 = -2$, which means

$$R_{1,1}(x) = \frac{-2 - x}{-2 + x} = \frac{2 + x}{2 - x}.$$

For $x \leq 0.78$, this is a better approximation of e^x than the truncated Taylor series $e^x = 1 + x + \frac{1}{2}x^2$, but for larger x , the pole of $R_{1,1}(x)$ at $x = 2$ causes the rational approximation to be poor.

In general, this approach does not guarantee a solution, as the system of $N+M+1$ linear equations obtained by zeroing the first $N+M+1$ coefficients of (2.53) may not have a solution [23, p. 222]. In that case, the Padé approximant does not exist.

The convergence of rational approximants under the uniform norm is discussed in [36]. Moreover, stronger convergence results are obtained if small denominator values are prohibited [37]. Several rational approximation algorithms, either minimizing the uniform norm or considering Padé approximation, are reviewed in [38].

Rational or polynomial approximation?

To conclude this chapter, we summarize the main advantages and disadvantages in using rational approximation instead of polynomial approximation. As mentioned earlier, the polynomial convergence results discussed in Sections 2.3.1–2.3.2, and the best approximation theory developed for general linear spaces, do not apply for rational functions. In general, the theory related to rational functions is not as well-established as that of the polynomial family.

On the other hand, the rational functions contain all polynomials as their subset. Even with considering splines, it can be proven that rational functions are not worse than splines for approximation under the uniform norm [39, Ch. 8]. Rational functions are able to model singularities (infinite function values $g(x)$ at some finite $x = a$), asymptotic growth, and functions whose derivatives become infinite or discontinuous at any point [23, p. 217]. Of course, the roots of the denominator can also cause undesired vertical asymptotes, if no effort is made in specifically constraining or eliminating them.

High-degree polynomial approximants may produce significant oscillations, as demonstrated earlier. Polynomials attain infinite values if and only if the input values x are infinite, so they are generally not suitable for enforcing asymptotic conditions or for modelling rational functions. Polynomials also have poor extrapolatory properties [40].

While polynomials are very efficient to handle computationally, rational functions are not much more computationally demanding. Also, the flexibility of the rational functions typically allow complicated structures to be modelled with rather low degrees in both the numerator and denominator, resulting in fewer coefficients, less oscillations and possibly a more stable model than with a high-degree polynomial required to model the same data [40]. However, it is not always obvious how to determine satisfactory values for the degrees N and M .

3. NOISE MODELS

Digital image acquisition is subject to various errors, some of which are signal-dependent and others signal-independent. Depending on the various physical phenomena underlying specific imaging modalities (e.g., digital cameras with CCD or CMOS sensors, magnetic resonance imaging, radar imaging, or X-ray computed tomography), the noise associated with the particular image formation process has specific characteristics. Proper noise modelling is an important part of processing images corrupted with noise, although sometimes concessions are made in order to describe a complex physical process with a simplified model, for instance for facilitating a convenient practical implementation. Here we introduce a few common noise models: Gaussian, Poisson, Poisson-Gaussian, Rice, and Rayleigh noise.

3.1 Gaussian noise

A traditional assumption in image processing is that the noise follows a Gaussian (i.e., normal) distribution. This model is well studied, simple to understand and implement, and in many cases its usage is justified, due to an important result of probability theory: the Central Limit Theorem (CLT). CLT, first defined generally by Lyapunov in 1901, but already known in some form to both de Moivre and Laplace in the previous century [41], states a condition under which the normalized sum of random variables converges to the standard normal distribution. This in turn suggests that if we have many different noise sources, we may try to collectively approximate them by a Gaussian distribution, instead of attempting to construct a highly complicated model taking into account each noise source separately. This condition of CLT (given in Lyapunov's form), for k independent random variables X_1, \dots, X_k with respective mean values μ_i and variances σ_i^2 , is defined as

$$\lim_{k \rightarrow \infty} \frac{1}{s_k^{2+\delta}} \sum_{i=1}^k E \{ |X_i - \mu_i|^{2+\delta} \} = 0, \quad (3.1)$$

where $s_k^2 = \sum_{i=1}^k \sigma_i^2$. If (3.1) holds for some $\delta > 0$, the normalized sum of the random variables converges in distribution to the standard normal distribution:

$$\frac{1}{s_k} \sum_{i=1}^k (X_i - \mu_i) \rightarrow \mathcal{N}(0, 1), \quad (3.2)$$

when $k \rightarrow \infty$.

Image processing applications often further assume that the noise is *additive white Gaussian noise* (AWGN). This means the noise is signal-independent, normally distributed, independent and identically distributed (the latter implying that σ_i is constant over the image), and has a flat power spectrum. We formulate the AWGN model as

$$z = y + \eta, \quad (3.3)$$

where z is the acquired noisy image, y is the underlying (unknown) noise-free image, and $\eta \sim \mathcal{N}(\mu, \sigma^2)$ is additive white Gaussian noise corrupting the ideal image y . However, this is often an overly simplified model of what really happens in the actual imaging chain. In particular, it does not take the signal-dependent shot noise (i.e., Poisson noise) into account in any way. Despite this significant omission, AWGN is a very commonly used assumption for image denoising algorithms, because it makes the algorithms generally more straightforward to design and implement than for signal-dependent models. That being said, there are also applications where the Gaussian model is more justified. As an example, the data obtained through low-dose computed tomography (CT) can be considered Gaussian, although with signal-dependent variance [42].

Concerning the basic properties of a Gaussian distributed variable $z \sim \mathcal{N}(\mu, \sigma^2)$, its probability density function (PDF) equals

$$P(z) := \frac{1}{\sigma} \phi\left(\frac{z - \mu}{\sigma}\right) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \quad (3.4)$$

where $\phi(\cdot)$ is the PDF of the standard normal distribution $\mathcal{N}(0, 1)$. Similarly, the cumulative distribution function (CDF) of z is

$$\Phi\left(\frac{z - \mu}{\sigma}\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^t e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt, \quad (3.5)$$

with $\Phi(\cdot)$ being the CDF of $\mathcal{N}(0, 1)$. By definition, the expectation and variance (i.e., second central moment) of z are μ and σ^2 , respectively. We will also utilize the skewness and (excess) kurtosis, which are the third and fourth standardized moments, respectively. Specifically, skewness is defined as

$$\text{skew}(z) := E\left\{\left(\frac{z - \mu}{\sigma}\right)^3\right\} = \frac{\mu_3}{\sigma^3}, \quad (3.6)$$

and kurtosis as

$$\text{kurt}(z) := E\left\{\left(\frac{z - \mu}{\sigma}\right)^4\right\} - 3 = \frac{\mu_4}{\sigma^4}, \quad (3.7)$$

where

$$\mu_n := E \{(z - E \{z\})^n\}$$

is the n -th central moment. Note that the subtraction of 3 in (3.7) normalizes the kurtosis of the Gaussian distribution to 0, hence the alternative name excess kurtosis.

3.2 Poisson noise

As mentioned, the AWGN model does not take into account any possible signal-dependent noise sources in the imaging chain. Most crucially, it ignores the inherent randomness related to the emission and sensing of photons, which in turn contributes to the randomness in the output of the imaging sensor. This signal-dependent noise is called shot noise, or Poisson noise. In particular, even for a light source of constant intensity, the number of emitted photons varies over time [43], and this fluctuation can be considered to follow the Poisson distribution [44].

In order to define the Poisson noise model, let $z_i, i = 1, \dots, n$, be the (noisy) pixel values obtained through an imaging device. We consider each pixel z_i to be a realization of an independent random Poisson variable, whose mean $y_i \geq 0$ is the underlying noise-free intensity value of that pixel. Then, the discrete Poisson probability of each z_i is

$$P(z_i | y_i) = \frac{y_i^{z_i} e^{-y_i}}{z_i!}. \quad (3.8)$$

Note that y_i is not only the mean of the Poisson variable z_i , but also its variance, in other words

$$E \{z_i | y_i\} = y_i = \text{var} \{z_i | y_i\}. \quad (3.9)$$

Formally, we define Poisson noise as the difference

$$\epsilon_i := z_i - E \{z_i | y_i\} = z_i - y_i, \quad (3.10)$$

which implies

$$E \{\epsilon_i | y_i\} = 0 \quad \text{and} \quad \text{var} \{\epsilon_i | y_i\} = \text{var} \{z_i | y_i\} = y_i.$$

This indicates that Poisson noise is indeed signal-dependent, as the noise variance is a function of the true intensity value. Specifically, the standard deviation of the noise ϵ_i equals $\sqrt{y_i}$. This also means that as the intensity value decreases (due to low lighting conditions or a short camera shutter time, for instance), the effect of Poisson noise increases, in other words the signal-to-noise ratio decreases.

3.3 Poisson-Gaussian noise

Since the AWGN model ignores signal-dependent noise sources, and the Poisson noise model ignores signal-independent noise sources, a natural development is to combine the two into a mixed Poisson-Gaussian noise model. In this model, the Poisson component accounts for the uncertainty related to shot noise, and the additive Gaussian component collectively accounts for all the signal-independent noise sources, as suggested by the CLT (3.2).

Similarly to the Poisson noise model, let \check{z}_i , $i = 1, \dots, n$ be the observed pixel intensity values. However, now each \check{z}_i is composed of a scaled Poisson component and an additive Gaussian component:

$$\check{z}_i = \alpha \rho_i + \eta_i, \quad (3.11)$$

where $\alpha > 0$ is a constant scaling term modelling the gain of the sensor, ρ_i is an independent random Poisson variable with an underlying expected value $y_i \geq 0$ (the underlying noise-free pixel value), and η_i is a random Gaussian variable with mean μ and standard deviation $\check{\sigma} \geq 0$. In short, $\rho_i \sim \mathcal{P}(y_i)$ and $\eta_i \sim \mathcal{N}(\mu, \check{\sigma}^2)$. The mean and the variance of \check{z}_i equal

$$E\{\check{z}_i | y_i, \alpha, \mu, \check{\sigma}\} = \alpha y_i + \mu, \quad \text{var}\{\check{z}_i | y_i, \alpha, \mu, \check{\sigma}\} = \alpha^2 y_i + \check{\sigma}^2.$$

Moreover, the Gaussian mean μ is often assumed to either be zero or incorporated in the noise-free signal as a constant shift, implying $\eta_i \sim \mathcal{N}(0, \check{\sigma}^2)$; this does not affect the variance of the data.

As shown in [7], it is sometimes convenient to reduce the number of parameters by simple affine mappings

$$z_i = \frac{\check{z}_i - \mu}{\alpha}, \quad \sigma = \frac{\check{\sigma}}{\alpha}, \quad (3.12)$$

resulting in a Poisson-Gaussian variable of the form

$$z_i = \rho_i + \eta_i, \quad (3.13)$$

where $\rho_i \sim \mathcal{P}(y_i)$ and $\eta_i \sim \mathcal{N}(0, \sigma^2)$. The PDF of z_i equals

$$P(z_i | y_i, \sigma) = \sum_{k=0}^{+\infty} \left(\frac{y_i^k e^{-y_i}}{k!} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_i-k)^2}{2\sigma^2}} \right), \quad (3.14)$$

Poisson-Gaussian noise is then defined as

$$\epsilon_i := z_i - E\{z_i | y_i, \sigma\}, \quad (3.15)$$

analogously to (3.10).

Alternative model, with an affine-variance Gaussian approximation

As an alternative, we present a slightly different Poisson-Gaussian model that was introduced in [1] for the modelling of raw image data. Furthermore, we then consider a Gaussian approximation of the latter. Let our observations be of the form

$$z_i = y_i + \rho_i + \eta_i, \quad (3.16)$$

where y_i is again the underlying noise-free pixel value, ρ_i is a Poisson noise component with

$$\chi(y_i + \rho_i) \sim \mathcal{P}(\chi y_i), \quad \chi > 0$$

and η_i is a Gaussian noise component with $\eta_i \sim \mathcal{N}(0, b)$, $b \geq 0$. Here, the parameter χ describes the quantum efficiency of the imaging sensor [1, p. 1738]. If we set $a := \chi^{-1}$, we have

$$E\{z_i \mid y_i, a, b\} = y_i, \quad \text{var}\{z_i \mid y_i, a, b\} = ay_i + b$$

for the variance and mean of z_i . In contrast with model (3.11), the mean of z_i now equals y_i without any scaling parameters. A gain parameter can also be incorporated in this model by setting $a := \chi^{-1}\tau$, where τ corresponds to the ISO sensitivity of the sensor. Moreover, a pedestal parameter can be incorporated in order to model the base charge always present in the sensor. In that case, b can also attain negative values, while the variance remains non-negative [1, p. 1738].

In order to simplify this model, parametrized by a and b , we can leverage the usual Gaussian approximation of a Poisson variable

$$\mathcal{P}(y) \approx \mathcal{N}(y, y),$$

whose accuracy increases as y increases. Specifically, for large enough y , it is very reasonable to consider the Poisson-Gaussian noise to be Gaussian with affine variance, i.e., following the distribution $\mathcal{N}(0, ay + b)$ [1, p. 1739].

3.4 Rice and Rayleigh noise

The Rice distribution was first introduced by the Bell Labs engineer S.O. Rice in [45; 46], in the context of analyzing noise that results in passing random noise through physical devices, in particular considering the shot noise in vacuum tubes or thermal noise (i.e., thermal agitation of electrons) in resistors. Also the noise in magnitude MRI images follows a Rice distribution [2]. Such an MRI image is

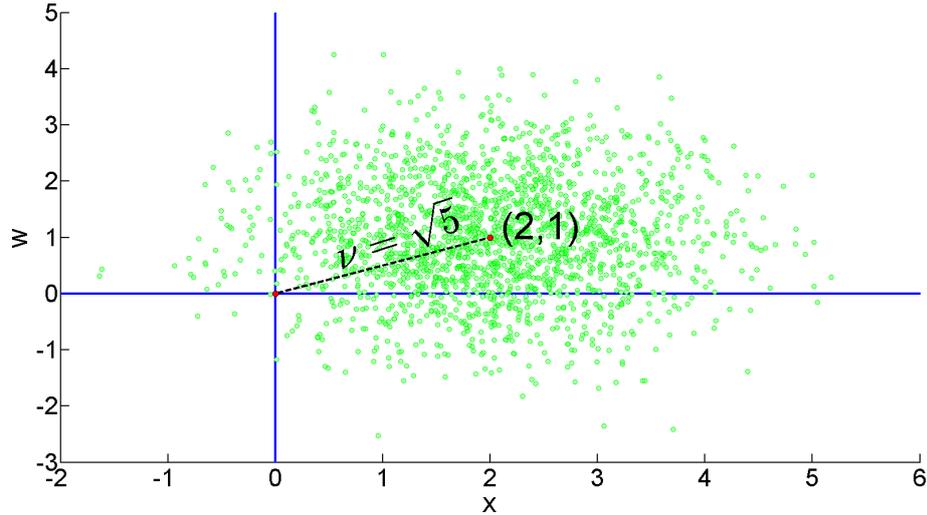


Figure 3.1: 2000 realizations (x_i, w_i) of independent random Gaussian variables $x \sim \mathcal{N}(2, 1)$, $w \sim \mathcal{N}(1, 1)$. The distance between each point (x_i, w_i) and the origin follows a Rice distribution $\mathcal{R}(\sqrt{5}, 1)$.

formed by computing the magnitude of each pixel based on real and imaginary signals obtained through a quadrature detector; the noise contaminating each of those signals is assumed to be independent Gaussian noise [2, p. 2].

As anticipated by the MRI image formation example, the Rice distribution arises from a complex Gaussian variable. Specifically, if both x and w follow a Gaussian distribution and are independent (i.e., x and w are jointly normal), and they have the same variance σ^2 , then

$$z_{x,w} := x + iw \quad (3.17)$$

follows a complex Gaussian distribution. This can be thought of as having a 2-D distribution in the Cartesian plane, centered at some distance $\nu \geq 0$ from the origin, where both coordinates x and w are independent realizations of a Gaussian distribution with variance σ^2 . Then, the magnitude $z := |z_{x,w}|$, in other words the distance of the complex random variable $z_{x,w}$ from the origin in the (x, w) plane, follows a Rice distribution: $z \sim \mathcal{R}(\nu, \sigma)$. Figure 3.1 illustrates this graphical representation for a Rice distributed variable $z \sim \mathcal{R}(\sqrt{5}, 1)$ centered at $(2, 1)$.

Explicitly, $z \sim \mathcal{R}(\nu, \sigma)$ is distributed according to the PDF

$$P(z | \nu, \sigma) := \frac{z}{\sigma^2} e^{-\frac{z^2 + \nu^2}{2\sigma^2}} I_0\left(\frac{z\nu}{\sigma^2}\right), \quad (3.18)$$

where I_k is the k -th order modified Bessel function of the first kind [47]:

$$I_k(z) = \sum_{m=0}^{\infty} \frac{1}{m!(k+m+1)!} \left(\frac{z}{2}\right)^{k+2m}, \quad k = 0, 1, \dots$$

The mean and the variance of a Rice distributed variable z are [6]

$$E\{z \mid \nu, \sigma\} = \sigma \sqrt{\frac{\pi}{2}} L_{1/2} \left(-\frac{\nu^2}{2\sigma^2} \right), \quad (3.19)$$

$$\text{var}\{z \mid \nu, \sigma\} = 2\sigma^2 + \nu^2 - \frac{\pi\sigma^2}{2} L_{1/2}^2 \left(-\frac{\nu^2}{2\sigma^2} \right), \quad (3.20)$$

where

$$L_{1/2}(x) = e^{\frac{x}{2}} \left[(1-x)I_0 \left(-\frac{x}{2} \right) - xI_1 \left(-\frac{x}{2} \right) \right]$$

is a Laguerre polynomial. As in the Poisson and Poisson-Gaussian cases, it is clear that the variance of a Rice distributed variable is signal-dependent.

Rayleigh noise is a notable special case of Rice noise, obtained when the complex 2-D distribution is centered at the origin in the (x, w) plane, meaning $\nu = 0$. Hence, according to (3.18), the PDF of a Rayleigh distributed variable $z \sim \mathcal{R}(0, \sigma)$ equals

$$P(z \mid 0, \sigma) = \frac{z}{\sigma^2} e^{-\frac{z^2}{2\sigma^2}}.$$

Further, the mean and the variance of a Rayleigh distributed variable z simplify from (3.19)–(3.20) into

$$E\{z \mid 0, \sigma\} = \sigma \sqrt{\frac{\pi}{2}}, \quad \text{var}\{z \mid 0, \sigma\} = \frac{(4 - \pi)\sigma^2}{2}.$$

The Rayleigh distribution is used, for instance, in analyzing the probability distribution of wind speed in a given location over a period of time [48], and in modelling the speckle noise arising in single-look synthetic-aperture radar imaging [49]. Both Rice and Rayleigh distributions are also useful in investigating urban radio propagation, where buildings and the atmosphere cause the radio signal to reflect, diffract and scatter, thus arriving at the receiver through multiple paths [50].

3.5 Clipping

In practice, imaging sensors have a limited dynamic range, because each pixel can only accumulate a certain amount of charge. Thus, values lower than the minimum possible value (underexposed pixels) will be equal to this minimum value in the sensor output, and similarly values higher than the maximum possible value (overexposed pixels) will be equal to the maximum value. This phenomenon is referred to as clipping, or censoring. Mathematically, this is often put in the form

$$\tilde{z} := \max\{0, \min\{z, 1\}\},$$

where \tilde{z} is the variable after clipping; the range can be limited to $[0, 1]$ without loss of generality. It is evident that the mean and the variance of \tilde{z} are generally not equal to those of z , so clipping introduces error with non-zero mean.

In general terms, according to (3.4), a random variable $z \sim \mathcal{N}(y, \sigma^2(y))$ with signal-dependent standard deviation $\sigma(y)$ has the PDF

$$P(z) = \frac{1}{\sigma(y)} \phi\left(\frac{z - \mu}{\sigma(y)}\right).$$

However, the generalized PDF of the *doubly censored* (clipped from both below and above) variable $\tilde{z} \in [0, 1]$ is divided into three parts, specifically

$$P(\tilde{z}) = \Phi\left(\frac{-y}{\sigma(y)}\right) \delta_0(\tilde{z}) + \frac{1}{\sigma(y)} \phi\left(\frac{\tilde{z} - \mu}{\sigma(y)}\right) \chi_{[0,1]} + \Phi\left(\frac{y-1}{\sigma(y)}\right) \delta_0(1 - \tilde{z}), \quad (3.21)$$

where $\chi_{[0,1]}$ is the characteristic function of the interval $[0, 1]$, and δ_0 is the Dirac delta impulse at 0 [51, p. 2631]. In other words, the generalized PDF has a continuous part representing the non-clipped range, and two discrete masses representing the clipped tails of the distribution. For other specific noise distributions, the formulation is analogous.

In most practical cases, it is reasonable to assume [1; 51] that if $z < 0$ is possible, then $z > 1$ is not. Then, only one of the impulses (either the first or the last term in (3.21)) will significantly differ from zero, and the variable is well approximated by a singly censored variable. The formulas for the means and variances of doubly and singly censored variables adhering to (3.21) are presented in [51].

4. VARIANCE STABILIZATION

As shown earlier, many noise models have the inherent property that the noise variance is not constant over the acquired data (e.g., an image), but varies with the expected value of the data; the noisy data is heteroskedastic. A *variance-stabilizing transformation* (VST) is a nonlinear transformation designed to remove this data-dependency for data adhering to a specific noise model, thus rendering the noise variance (ideally) constant throughout the data. More specifically, the transformed data can be assumed to have an approximately Gaussian noise distribution with a known constant variance, usually scaled to unity. That is, a VST is typically not only stabilizing, but also normalizing to some extent. This method allows the transformed data to be further processed with algorithms designed for additive white Gaussian noise, which is often easier and more practical than finding or even designing algorithms that are specifically tailored for each noise model. Considering image denoising for instance, the AWGN model is well studied, with many efficient algorithms (e.g., [52; 53]) to choose from. Hence, we may apply a VST to heteroskedastic data, and then denoise it with a state-of-the-art algorithm designed for the removal of AWGN. Finally, we need to apply a suitable inverse of the VST to the processed data, in order to return it to the original range.

In Section 4.1, we present a traditional heuristic for constructing VSTs, and comment on various important general results regarding variance stabilization. Sections 4.2–4.3 review some of the existing VSTs designed for Poisson, Poisson-Gaussian, Rice and Rayleigh distributed data. Finally, Section 4.4 concisely elaborates on the importance of proper inverse transformations; the author has written extensively about the inversion in [3].

4.1 Development of VSTs

First, let us consider any noisy data z without specifying a noise distribution. An ideal VST $f(z)$ would be a transformation, for which $\text{var}\{f(z) | y\} = c$, where $c > 0$ is a constant. Since the variance of $f(z)$ is constant, it is clearly independent of the expected value $E\{z | y\} =: \mu(y)$. Without loss of generality, we assume $c = 1$ in the rest of this thesis. A traditional starting point in finding f for a specific distribution is to compute the integral

$$f(z) = \int_z \frac{1}{\sigma(y)} d\mu(y), \quad (4.1)$$

where $\text{std}\{z | y\} := \sigma(y)$. This heuristic has been used at least since 1935 [54], although given its simplicity, it has likely been independently discovered by various people. Usually it does not yield a particularly good stabilizer in itself, but it can give an idea of the types of transformations to investigate further.

Results concerning the stabilization of general distribution families are famously presented in [55; 56; 57]. More recently, the optimization of VSTs has been formulated as an explicit optimization problem [10; 11], resulting in nonparametric state-of-the-art stabilizers for many common distribution families. In this thesis, we develop a similar approach for optimizing parametric stabilizers, specifically polynomial and rational VSTs, concentrating on the Poisson-Gaussian and Rice families. For more details on the history and development of variance stabilization, we refer to [3, pp. 12–15].

4.2 Poisson and Poisson-Gaussian noise

For Poisson corrupted data z , we have $\mu(y) = y$ and $\sigma(y) = \sqrt{y}$. Hence, (4.1) yields a stabilizer $f(z) = 2\sqrt{z}$. Many refinements of this square root transformation have been proposed, such as the Bartlett transformation $f(z) = 2\sqrt{z + \frac{1}{2}}$ [58], the Anscombe transformation $f(z) = 2\sqrt{z + \frac{3}{8}}$ [4], and the Freeman-Tukey transformation $f(z) = \sqrt{z} + \sqrt{z+1}$ [59]. The stabilization accuracy of these four transformations and a state-of-the-art nonparametric optimized transformation [11], for the interval $[0, 5]$, is visualized in Figure 4.1. While the standard deviation approaches unity *asymptotically* (i.e., as $y \rightarrow \infty$), different stabilizers converge at different speeds. For instance, the asymptotic stabilized variance is of the order $1 + \mathcal{O}(y^{-2})$ for the Anscombe transformation¹, but only $1 + \mathcal{O}(y^{-1})$ for the simple heuristic transformation [3, p. 15]. In any case, the square root transformations are unable to provide particularly good stabilization for low intensity values y . This inaccuracy may in turn lead to mediocre performance in practical applications dealing with low-intensity Poisson or Poisson-Gaussian data [7; 8; 9]. Hence, investigating the optimization of VSTs is warranted, despite the fact that it is impossible to achieve exact stabilization for Poisson and Poisson-Gaussian data [55; 56].

For Poisson-Gaussian data \tilde{z} following the model (3.11), a common choice of VST is the generalized Anscombe transformation (GAT) [5]

$$f_{\alpha, \mu, \tilde{\sigma}}(\tilde{z}) = \begin{cases} \frac{2}{\alpha} \sqrt{\alpha \tilde{z} + \frac{3}{8} \alpha^2 + \tilde{\sigma}^2} - \alpha \mu, & \tilde{z} > -\frac{3}{8} \alpha - \frac{\tilde{\sigma}^2}{\alpha} + \mu \\ 0, & \tilde{z} \leq -\frac{3}{8} \alpha - \frac{\tilde{\sigma}^2}{\alpha} + \mu \end{cases}, \quad (4.2)$$

¹The addend $\frac{3}{8}$ in the Anscombe transformation is obtained by zeroing the first-order term in the Taylor expansion of the stabilized variance, resulting in $\text{var}\{f(z) | y\} = 1 + \frac{1}{16y^2}$ for $y \rightarrow \infty$ [3, p. 15].

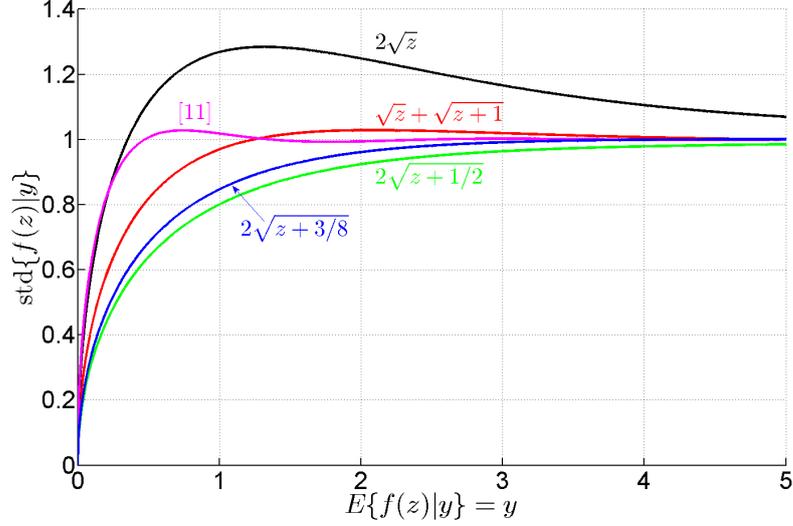


Figure 4.1: Standard deviation of the stabilized Poisson variable $f(z)$ for the heuristic transformation $f(z) = 2\sqrt{z}$, the Bartlett transformation $f(z) = 2\sqrt{z + 1/2}$, the Anscombe transformation $f(z) = 2\sqrt{z + \frac{3}{8}}$, the Freeman-Tukey transformation $f(z) = \sqrt{z} + \sqrt{z + 1}$, and a state-of-the-art nonparametric optimized transformation [11] (nonmonotone, direct search).

which in fact is a family of transformations, parametrized by α , μ and $\check{\sigma}$. Observe that by setting $\alpha = 1$, $\mu = 0$ and $\check{\sigma} = 0$ we obtain the pure Poisson case, and the GAT coincides with the Anscombe transformation.

If the affine mappings (3.12) are applied, the GAT for the model (3.13) simplifies into

$$f_{\sigma}(z) = \begin{cases} 2\sqrt{z + \frac{3}{8} + \sigma^2}, & z > -\frac{3}{8} - \sigma^2 \\ 0, & z \leq -\frac{3}{8} - \sigma^2 \end{cases}, \quad (4.3)$$

which is more conveniently a family of transformations parametrized only by σ . Thus, in the experiments presented in this thesis, the GAT refers to (4.3), as we can always perform the affine mapping beforehand for general Poisson-Gaussian data \check{z} with $\alpha \neq 1$ and $\mu \neq 0$ (and the corresponding affine inverse mapping as the very final step after stabilizing z , denoising or otherwise processing the stabilized data $f_{\sigma}(z)$, and applying the inverse VST).

Owing to the properties of the Anscombe transformation, the GAT also achieves an asymptotic convergence rate of $\text{var}\{f_{\sigma}(z) | y\} = 1 + \mathcal{O}(y^{-2})$ for $y \rightarrow \infty$. On the other hand, for small values of σ (i.e., close to the pure Poisson distribution) it has similar limitations in the stabilization accuracy for low intensities, as shown in Figure 4.2 for the values $\sigma = 0.01, 1, 2, 3$ ($\mu = 0$ and $\alpha = 1$).

As for the alternative model parametrized by a and b , we can reformulate the

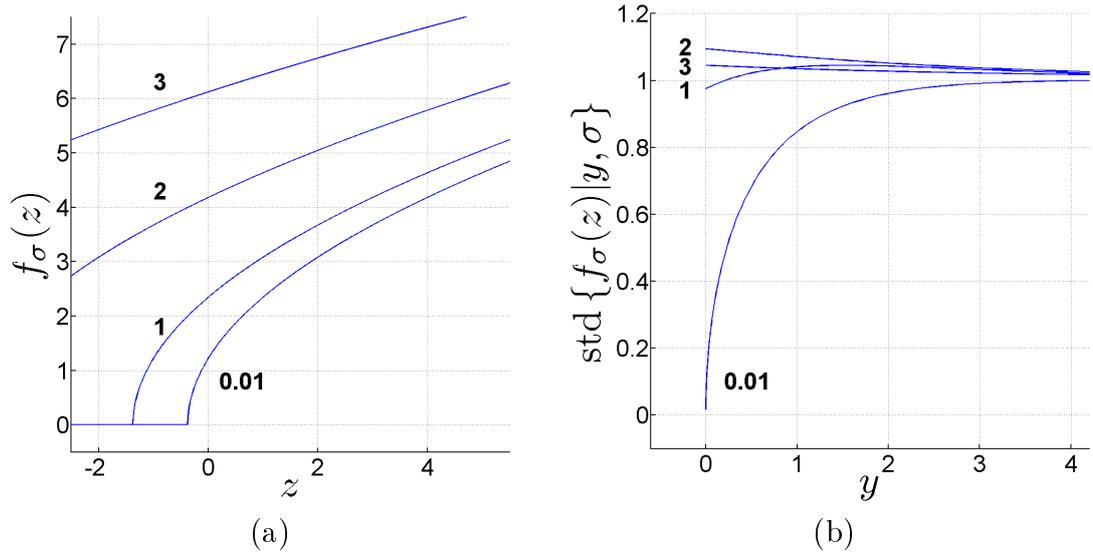


Figure 4.2: Generalized Anscombe transformation (4.3) for $\sigma = 0.01, 1, 2, 3$. (a) Forward transformations $f_\sigma(z)$. (b) Stabilized standard deviations $\text{std} \{f_\sigma(z) | y, \sigma\}$.

GAT (4.2) into

$$f_{a,b}(z) = \begin{cases} \frac{2}{a} \sqrt{az + \frac{3}{8}a^2 + b}, & z > -\frac{3}{8}a - \frac{b}{a} \\ 0, & z \leq -\frac{3}{8}a - \frac{b}{a} \end{cases} \quad (4.4)$$

in order to stabilize z following the model (3.16).

4.3 Rice and Rayleigh noise

For Rice distributed data $z \sim \mathcal{R}(\nu, \sigma)$, the variance (3.20) can be asymptotically (i.e., for large ν) expressed as a function of the mean μ (3.19), with the help of asymptotic expansions of the Bessel functions I_n . Specifically, this yields [6]

$$\text{var} \{z | \mu, \sigma\} = \sigma^2 \left(1 - \frac{\sigma^2}{2\mu^2} + \mathcal{O}(\mu^{-4}) \right). \quad (4.5)$$

Then, if we ignore the $\mathcal{O}(\mu^{-4})$ term, the heuristic (4.1) produces an asymptotic stabilizer

$$\begin{aligned} f_\sigma(z) &= \frac{1}{\sigma} \int_z \frac{1}{\sqrt{1 - \frac{\sigma^2}{2\mu^2}}} d\mu \\ &= \frac{1}{\sigma} \int_z \frac{\mu}{\sqrt{\mu^2 - \frac{\sigma^2}{2}}} d\mu \\ &= \frac{1}{\sigma} \sqrt{z^2 - \frac{\sigma^2}{2}} + c, \end{aligned}$$

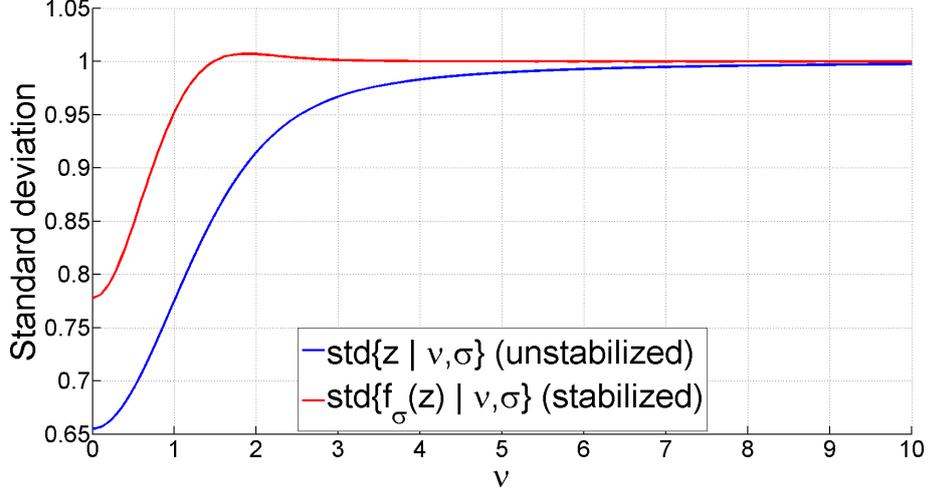


Figure 4.3: Unstabilized standard deviation $\text{std}\{z \mid \nu, 1\}$ and the asymptotically stabilized standard deviation $\text{std}\{f_\sigma(z) \mid \nu, 1\}$ for a Rice variable $z \sim \mathcal{R}(\nu, 1)$.

where $c \in \mathbb{R}$ is an arbitrary constant of integration. Hence, we define the asymptotic Rice VST as

$$f_\sigma(z) = \begin{cases} \sqrt{\frac{z^2}{\sigma^2} - \frac{1}{2}} + c, & z \geq \frac{\sigma}{\sqrt{2}} \\ c, & z < \frac{\sigma}{\sqrt{2}} \end{cases}, \quad (4.6)$$

as in [6].

Without loss of generality, we may even assume that σ is fixed (say, $\sigma = 1$), as the other cases $\sigma \neq 1$ can be handled through simple scaling. Specifically, if $z \sim \mathcal{R}(\nu, \sigma)$, then it follows from the definition (3.17) that $\lambda z \sim \mathcal{R}(\lambda\nu, \sigma)$ for any constant $\lambda > 0$.

Figure 4.3 shows, for the case $z \sim \mathcal{R}(\nu, 1)$, the unstabilized standard deviation $\text{std}\{z \mid \nu, 1\}$ and the asymptotically stabilized standard deviation $\text{std}\{f_\sigma(z) \mid \nu, 1\}$. We see that when $\sigma = 1$, also the unstabilized standard deviation converges to unity, albeit much slower than the stabilized standard deviation. However, $\text{std}\{z \mid \nu, \sigma\}$ does not converge to unity in the general case $\sigma \neq 1$, as is evident from (4.5).

As for the special case of the Rayleigh distribution, i.e., $z \sim \mathcal{R}(0, \sigma)$, the logarithmic VST

$$f(z) = 2\sqrt{6}\pi^{-1} \ln(z)$$

stabilizes the noise variance perfectly to $\text{var}\{f(z) \mid \sigma\} = 1$ for all $\sigma > 0$ [60]. However, the unboundedness of this function may be problematic, so one may instead use the nonparametric optimization scheme [11] to find an optimized approximate stabilizer with a bounded derivative, as was done in [49].

4.4 Inverse transformations

As explained earlier, using variance stabilization to process heteroskedastic data includes three steps, last of which is the application of an inverse VST. In particular, a denoising algorithm applied to stabilized data $f_\theta(z)$, where θ denotes all the parameters of the distribution (e.g., $\theta = [\alpha, \mu, \sigma]$), attempts to estimate the mean $E\{f_\theta(z) | y, \theta\}$. However, since the forward VST f_θ is necessarily nonlinear, we generally have

$$E\{f_\theta(z) | y, \theta\} \neq f_\theta(E\{z | y, \theta\}), \quad (4.7)$$

which further means

$$f_\theta^{-1}(E\{f_\theta(z) | y, \theta\}) \neq E\{z | y, \theta\}. \quad (4.8)$$

Thus, using a VST f_θ to stabilize z , denoising the stabilized data, and finally applying the algebraic inverse of f_θ does not generally produce the desired estimate $E\{z | y, \theta\}$ of the unstabilized data; the obtained estimate is biased. The traditional approach, especially for Poisson noise [4], has been to construct an adjusted inverse that provides asymptotic unbiasedness for large values of y . However, this approach does not mitigate the problem of bias for low values of y , unlike the more recent concept of *exact unbiased inversion* discussed for instance in [51; 49; 61; 6; 62; 3]. An exact unbiased inverse is equally applicable for all values of y , because it is defined as the mapping

$$E\{f_\theta(z) | y, \theta\} \rightarrow E\{z | y, \theta\}. \quad (4.9)$$

This mapping can be constructed by numerically evaluating $E\{f_\theta(z) | y, \theta\}$ and $E\{z | y, \theta\}$ for a grid of values y , and using them to interpolate the mapping for arbitrary values within the grid. For large enough y , interpolation can be supplanted by an asymptotically unbiased inverse, as the latter becomes accurate enough by design. For the Anscombe transformation and the GAT, we have also constructed closed-form approximations of their exact unbiased inverses in [62] and [7], respectively.

Note also that traditionally the forward VST f_θ is required to be monotonic, in order to guarantee the existence of its algebraic inverse f_θ^{-1} . However, using an exact unbiased inverse means this requirement can be relaxed: f_θ can be nonmonotonic, as long as the mapping $E\{z | y, \theta\} \rightarrow E\{f_\theta(z) | y, \theta\}$ is monotonic, thus guaranteeing that the exact unbiased inverse (4.9) exists.

5. OPTIMIZATION OF VSTS USING POLYNOMIALS

The parametric VSTs presented in Chapter 4 have an inherent limitation in their stabilization accuracy; this is most notable in the low-intensity range, due to the asymptotic construction of the VSTs. In order to address this deficiency, we investigate the optimization of polynomial VSTs in Chapter 5, and rational VSTs in Chapter 6. Our aim is to describe a general-purpose framework for optimizing various types of VSTs (such as polynomials and rational functions) for various noise distributions, with the core optimization procedure remaining the same regardless of the type of VST or the noise distribution. First, we explore the more straightforward case of optimizing polynomial VSTs, as it does not require distribution-specific considerations in the way that optimizing rational VSTs does. Specifically, using rational VSTs enables us to set asymptotic constraints, which we need to tailor for each distribution.

As mentioned, the optimization of VSTs was recently considered as an explicit optimization problem in [11]. It results in highly effective nonparametric VSTs, whereas we are more interested in optimizing the performance of parametric VSTs. However, we can adopt the same general approach: the goal of the optimization is to minimize the discrepancy between the stabilized variance and a target function, which in this case is a constant function $g(x) \equiv 1$. In other words, we want the function $\text{var}\{f_\theta(z) \mid y, \theta\}$ to approximate a constant function 1 as closely as possible, according to a certain metric. Let us start with the basic least squares approach, and consider a function $f_\theta(z)$ determined through the coefficients p_i , $i = 0, \dots, n$, which are to be optimized. In order to emphasize this dependency on the coefficients p_i , we employ a shorthand notation $f_p(z)$. Thus, in this section, $f_p(z)$ is a polynomial of degree n . Then, we can formulate the coefficient optimization as the minimization of an integral cost functional over all intensity values y :

$$\arg \min_{\{p_i\}_{i=0}^n} \int_{\mathbb{R}^+} (\text{var}\{f_p(z) \mid y, \theta\} - 1)^2 dy. \quad (5.1)$$

However, there are various additional constraints we want to impose as well. First, we add an option to penalize overshoot in the stabilized variance differently than undershoot. For instance, in image denoising, overshoot may lead to too light denois-

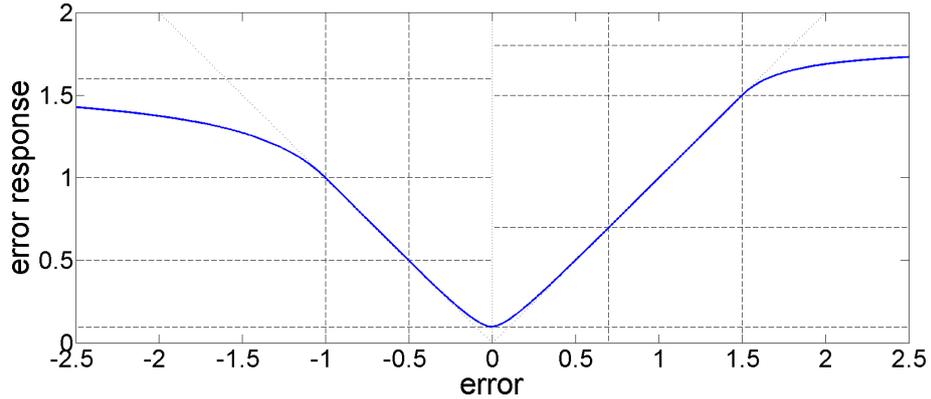


Figure 5.1: A nonlinear function Γ used in regulating the error $\text{var}\{f_p(z) | y, \theta\} - 1$ in the stabilized variance. The left half corresponds to the undershoot of variance, and the right half to the overshoot of variance. This is an example curve created for visualization, not the one used in our experiments.

ing (which can leave visible noise in otherwise smooth areas, and also produce artifacts), whereas undershoot typically leads to a visually more pleasing oversmoothing. Thus, we split the absolute difference $|\text{var}\{f_p(z) | y, \theta\} - 1|$ into two terms, based on whether the difference is positive or negative. Consequentially, we also ignore the second power in (5.1).

Second, we regulate the errors in the stabilized variance in such a way that very large errors will have only a limited impact on the cost functional, and contrarily, very small errors will still have some impact. In practice, this is done by passing the error term through a nonlinear response curve Γ . Figure 5.1 shows an example curve Γ demonstrating the concept.

Third, as explained in the previous section, we do not need to enforce the monotonicity of $f_p(z)$. On the other hand, we do want the mapping

$$h : E\{z | y, \theta\} \rightarrow E\{f_p(z) | y, \theta\}$$

to be monotonic (in practice, strictly increasing), in order to guarantee the existence of the exact unbiased inverse of $f_p(z)$. Thus, we add a term penalizing any h with even a single point with a non-positive derivative, so that they will be considered inferior to a strictly increasing mapping h .

Fourth, we wish to not only stabilize the data, but also normalize (i.e., Gaussianize) it at least to some extent. Hence, we add terms penalizing large absolute values of skewness (3.6) and kurtosis (3.7).

Formulating all of the above into (5.1), we get a cost functional

$$\begin{aligned}
C(p) &= \int_{\mathbb{R}^+} [\Gamma(\varepsilon_o(p)) + \Gamma(\varepsilon_u(p))] \, dy + U \cdot \sum_{k: h'(k) \leq 0} (-h'(k) + 1) \\
&\quad + \lambda_s \int_{\mathbb{R}^+} |\text{skew}(f_p(z))| \, dy + \lambda_k \int_{\mathbb{R}^+} |\text{kurt}(f_p(z))| \, dy, \tag{5.2}
\end{aligned}$$

where the (weighted) errors in the stabilized variance are

$$\begin{aligned}
\varepsilon_o(p) &:= \lambda_o \cdot \max\{0, (\text{var}\{f_p(z) \mid y, \theta\} - 1)\} \\
\varepsilon_u(p) &:= \lambda_u \cdot \max\{0, (1 - \text{var}\{f_p(z) \mid y, \theta\})\},
\end{aligned}$$

and the non-negative constants λ_o , λ_u , λ_s and λ_k are weights for the amount of penalization associated with overshoot, undershoot, skew and kurtosis, respectively. Moreover, the constant U is defined as the maximum attainable value of the first integral in (5.2).

Hence, by minimizing the cost functional (5.2), we obtain the optimized polynomial coefficients p_i , $i = 0, \dots, n$ that define the n -th degree optimized polynomial VST. Practical implementation details regarding the minimization are discussed in Section 7.

6. OPTIMIZATION OF VSTS USING RATIONAL FUNCTIONS

Having discussed the optimization of polynomial VSTs, we can now extend the presented method to encompass the optimization of rational functions. However, since rational functions enable us to enforce asymptotic constraints on the VST, we need to formulate these constraints separately for each noise distribution. In particular, we consider the Poisson-Gaussian and Rice distributions, as the Poisson and Rayleigh distributions are special cases of the two former, respectively.

6.1 Poisson-Gaussian noise

In this section, our main focus is on the Poisson-Gaussian model (3.13). As explained in Section 3.3, we need to only consider the one-parameter family of VSTs parametrized by σ , assuming $\mu = 0$ and $\alpha = 1$, as the general model (3.11) can then be addressed through the affine mappings (3.12). The treatment for the alternative Poisson-Gaussian model (3.16) parametrized by a and b proceeds similarly, so the results for the latter are briefly summarized in the end.

Let $R(z) = \frac{P(z)}{Q(z)}$ be defined as in (2.46). For our purposes, we want the optimized rational VST f_σ to approach the GAT (4.3) asymptotically. In particular, this guarantees that the optimized VST always attains good asymptotic stabilization, as can be shown through simple calculus following the results in [63]. Hence, due to the GAT being a square root type of transformation, we will in fact optimize a rational function inside the typical square root transformation, and define

$$f_\sigma(z) := 2\sqrt{\frac{P(z)}{Q(z)}} = 2\sqrt{\frac{\sum_{i=0}^N p_i z^i}{\sum_{i=0}^M q_i z^i}}.$$

As noted in Section 2.2, we can scale the coefficients so that one of them becomes unitary. Thus, we assume that $q_0 = 1$. Then, our task is to optimize the remaining $N + M + 1$ coefficients p_i , $i = 0, \dots, N$ and q_i , $i = 1, \dots, M$ for each fixed value of σ .

Regarding the asymptotic constraint, in practice we ensure that

$$\frac{P(z)}{Q(z)} - \left(z + \frac{3}{8} + \sigma^2\right) \rightarrow 0 \quad \text{as } z \rightarrow +\infty \quad (6.1)$$

at a rate of $\mathcal{O}(z^{-1})$. First of all, (6.1) clearly implies that we must have $M = N - 1$ and $q_M = p_N$, so that the highest-order term will vanish as $z \rightarrow +\infty$. In other words, a precondition of attaining the desired asymptotic stabilization is that the degree of the numerator $P(z)$ is one more than the degree of the denominator $Q(z)$. After including these two requirements, condition (6.1) gets the form

$$\frac{p_N z^N + \dots + p_1 z + p_0 - (z + \frac{3}{8} + \sigma^2)(p_N z^{N-1} + q_{N-2} z^{N-2} + \dots + q_1 z + 1)}{p_N z^{N-1} + q_{N-2} z^{N-2} + \dots + q_1 z + 1} \rightarrow 0$$

as $z \rightarrow +\infty$. The N -th order terms cancel each other as intended, so after simplification, and denoting $s := \frac{3}{8} + \sigma^2$ for compactness, we obtain the condition

$$\frac{(-s p_N + p_{N-1} - q_{N-2}) z^{N-1} + \sum_{i=1}^{N-2} (p_i - s q_i - q_{i-1}) z^i - s + p_0}{p_N z^{N-1} + q_{N-2} z^{N-2} + \dots + q_1 z + 1} \rightarrow 0 \quad (6.2)$$

as $z \rightarrow +\infty$. Thus, we satisfy the convergence at a rate of $\mathcal{O}(z^{-1})$ by also setting the coefficient of z^{N-1} in the numerator of (6.2) to zero, in other words $q_{N-2} = p_{N-1} - (\frac{3}{8} + \sigma^2) p_N$. Consequentially, we can define

$$f_\sigma(z) = 2 \sqrt{\frac{p_N z^N + \dots + p_1 z + p_0}{p_N z^{N-1} + [p_{N-1} - (\frac{3}{8} + \sigma^2) p_N] z^{N-2} + q_{N-3} z^{N-3} + \dots + q_1 z + 1}}, \quad (6.3)$$

with $2N - 2$ coefficients p_0, \dots, p_N and q_1, \dots, q_{N-3} to be optimized.

Let us look at a practical example with $N = 3$ and $M = 2$, so that our rational function $P(z)/Q(z)$ has a cubic numerator and a quadratic denominator. Then, the VST (6.3) gets the form

$$f_\sigma(z) = 2 \sqrt{\frac{p_3 z^3 + p_2 z^2 + p_1 z + p_0}{p_3 z^2 + [p_2 - (\frac{3}{8} + \sigma^2) p_3] z + 1}}, \quad (6.4)$$

which depends solely on the four coefficients p_0, p_1, p_2, p_3 .

Assuming a fixed value of σ and the corresponding VST f_σ , we proceed to optimize the coefficients p_0, p_1, p_2, p_3 in much the same way as when optimizing the polynomial VST: by minimizing the integral stabilization cost functional (5.2). However, in order to avoid degenerate solutions, we supplement the cost functional by adding terms ensuring the following restrictions:

- (I) The cubic numerator $P(z)$ has only one (real) root.
- (II) The quadratic denominator $Q(z)$ is strictly positive, and its minimum value is not very close to zero.
- (III) The denominator is indeed quadratic and the numerator cubic: $p_3 \neq 0$.

This way, we also avoid all the complications that could arise due to f_σ having poles.

For the model (3.16) parametrized by a and b , we begin by setting

$$f_{a,b}(z) := \frac{2}{\sqrt{a}} \sqrt{\frac{P(z)}{Q(z)}}.$$

Note the inclusion of the multiplier $\frac{2}{\sqrt{a}} = \sqrt{a} \cdot \frac{2}{a}$, compensating for the multiplier $\frac{2}{a}$ in the GAT (4.4) for this model. Hence, we want to satisfy the asymptotic constraint

$$\frac{aP(z)}{Q(z)} - \left(az + \frac{3}{8}a^2 + b \right) \rightarrow 0 \quad \text{as } z \rightarrow +\infty.$$

Proceeding in the same way as for the other model, we obtain the general constraints

$$M = N - 1, \quad q_M = p_N \quad \text{and} \quad q_{N-2} = p_{N-1} - \left(\frac{3}{8}a + \frac{b}{a} \right) p_N.$$

For the case $N = 3$ and $M = 2$, this yields the VST

$$f_{a,b}(z) = \frac{2}{\sqrt{a}} \sqrt{\frac{p_3 z^3 + p_2 z^2 + p_1 z + p_0}{p_3 z^2 + [p_2 - (\frac{3}{8}a + \frac{b}{a}) p_3] z + 1}}. \quad (6.5)$$

Again, assuming fixed values of a and b , the VST depends only on the four coefficients p_0, p_1, p_2, p_3 , to be optimized as above.

6.2 Rice noise

The case of optimizing a rational VST for Rice noise is rather similar to the Poisson-Gaussian case, yet with some important differences. Recall that we can optimize the VST f_σ assuming a fixed value of σ (say, $\sigma = 1$), as the cases $\sigma \neq 1$ can be addressed through scaling. Nevertheless, in the discussion below, we consider a general σ for completeness. Let us start with the definition

$$f_\sigma(z) := \sqrt{\frac{P(z)}{Q(z)}} = \sqrt{\frac{\sum_{i=0}^N p_i z^i}{\sum_{i=0}^M q_i z^i}},$$

and again assume $q_0 = 1$. As in the Poisson-Gaussian case, we want f_σ to asymptotically converge to the asymptotic Rice VST (4.6):

$$\frac{P(z)}{Q(z)} - \left(\frac{z^2}{\sigma^2} - \frac{1}{2} \right) \rightarrow 0 \quad \text{as } z \rightarrow +\infty, \quad (6.6)$$

at a rate of $\mathcal{O}(z^{-1})$, similarly motivated by the results in [63]. The constant c in (4.6) has been set to zero; note that a constant shift in the VST does not affect the stabilization. This time, we immediately see that the convergence enforces at least $M = N - 2$ and $q_M = \sigma^2 p_N$. Thus, the degree of the numerator $P(z)$ must be two more than that of the denominator $Q(z)$. Taking these two requirements into account, we expand and simplify (6.6), just like in the previous section. As a result, we get a condition that

$$\sqrt{\frac{(p_{N-1} - \frac{q_{N-3}}{\sigma^2})z^{N-1} + (\frac{\sigma^2}{2}p_N + p_{N-2} - \frac{q_{N-4}}{\sigma^2})z^{N-2} + \dots + (p_1 + \frac{q_1}{2})z + p_0 + \frac{1}{2}}{\sigma^2 p_N z^{N-2} + q_{N-3} z^{N-3} + \dots + q_1 z + 1}} \quad (6.7)$$

should approach 0 as $z \rightarrow +\infty$. Convergence at a rate of $\mathcal{O}(z^{-1})$ is thus achieved by setting

$$M = N - 2, \quad q_M = \sigma^2 p_N, \quad p_{N-1} = \frac{q_{N-3}}{\sigma^2} \quad \text{and} \quad p_{N-2} = \frac{q_{N-4}}{\sigma^2} - \frac{\sigma^2}{2} p_N, \quad (6.8)$$

assuming coefficients with negative indices to be zero.

Let us look at two cases in more detail: first, having $N = 4$ and $M = 2$, and second, having $N = 3$ and $M = 1$. Using the constraints (6.8), the corresponding VSTs assume the forms

$$f_\sigma(z) = \sqrt{\frac{p_4 z^4 + \frac{1}{\sigma^2} z^3 + (\frac{1}{\sigma^2} - \frac{\sigma^2}{2} p_4) z^2 + p_1 z + p_0}{\sigma^2 p_4 z^2 + q_1 z + 1}}, \quad (N = 4, M = 2), \quad (6.9)$$

$$f_\sigma(z) = \sqrt{\frac{p_3 z^3 + \frac{1}{\sigma^2} z^2 - \frac{\sigma^2}{2} p_3 z + p_0}{\sigma^2 p_3 z + 1}}, \quad (N = 3, M = 1). \quad (6.10)$$

Hence, we have four coefficients to optimize for the 4/2 degree VST, and two for the 3/1 degree VST. Once again, that is done by minimizing the cost functional (5.2), accompanied by terms penalizing degenerate solutions, such as the denominator attaining zero, or the highest-order coefficient being zero.

7. EXPERIMENTAL RESULTS

In this chapter, we first discuss the practical implementation details related to the optimization of VSTs. Then, we present results for some example cases, considering the Poisson-Gaussian model (3.11), the affine-variance Gaussian approximation of the Poisson-Gaussian model (3.16), and the Rice distribution, with both unclipped and clipped data.

7.1 Implementation details

In Chapters 5–6, we described a framework for the optimization of polynomial and rational VSTs. In practice, we implement this as a general-purpose modular Matlab toolbox, with the goal of keeping the core optimization routines largely agnostic of the used distribution and the exact form of the VST. Even though we experiment explicitly with polynomials and rational functions for stabilizing Poisson-Gaussian and Rice data, the toolbox is built in such a way that adding other types of VSTs and other noise distributions is relatively straightforward. Essentially, each distribution is contained in one module, where the user sets the desired noise parameters, generates the probability distributions with a desired precision, specifies a parametric form for the VST to be optimized, and optionally sets distribution-specific optimization constraints (such as penalizing the denominator roots of a rational VST). These are then relayed to the optimization module, which prepends the distribution-specific constraints to the general cost functional (5.2) and iteratively optimizes the coefficients. In addition, it computes the exact unbiased inverse of the optimized VST, and a polynomial approximation of the inverse.

Let us elaborate on some of the implementation details related to the optimization itself. First, instead of integrating (5.2) over the whole \mathbb{R}^+ , we consider a range of y up to some maximum value, beyond which the asymptotic inverse is already accurate. For unclipped data z , we set it to $y_{\max} = 25$, and for data \tilde{z} clipped to interval $[C_1, C_2]$, we can decrease this upper bound, while ensuring that the clipping of noise is fully developed at y_{\max} . Keep in mind that as the mean value y (or ν) increases, the Poisson-Gaussian and Rice distributions are becoming more and more Gaussian, so restricting the range is warranted.

Moreover, as pointed out earlier, obtaining good stabilization accuracy at the lowest intensities is essential, as even the asymptotic stabilizers achieve accurate

stabilization for high intensities. Thus, we employ a quadratic spacing of the nodes y_i (i.e., the values $\sqrt{y_i}$ are uniformly spaced), in order to have more nodes for the low intensities than for the higher intensities. In practice, the integrals in the cost functional are numerically evaluated by trapezoidal integration. In our experiments, we use 120 nodes y_i .

Before the actual iterative coefficient optimization takes place, we need to initialize the VST. For optimizing a polynomial VST of degree n , this is done by fitting the $n + 1$ coefficients to either the GAT or the asymptotic Rice VST, depending on the distribution. In practice, the Matlab command `polyfit` does this by constructing the Vandermonde matrix V (2.4), and then obtaining the coefficients p by solving the associated least squares problem $Vp = y$. As discussed in Section 2.1.1, the Vandermonde matrix is often ill-conditioned, which makes the direct computation of its inverse (or pseudoinverse) numerically unstable. Hence, `polyfit` computes the QR decomposition $V = QR$, where Q is an orthonormal matrix: $Q^T = Q^{-1}$. Then, the least squares formulation yields $p = R^{-1}Q^T y$. In order to initialize a rational VST, we can choose the free coefficients rather arbitrarily, compute the remaining coefficients (which enforce the asymptotic constraints) based on them, and rely on the iterative optimization to take care of the rest; in practice, we have not noticed such initialization to cause any adverse effects in the results.

The coefficient optimization, in other words minimizing the cost functional, is done by iteratively applying the Nelder-Mead algorithm. It is a classical derivative-free optimization method [64, Ch. 8], which uses simplices to gradually move towards the minimum of the function. As long as the value of the cost functional keeps decreasing, we restart a new iteration of the Nelder-Mead, using the coefficients from the previous optimization round as initial values. However, if we do not obtain a lower cost functional value than before, we perturb the coefficients before starting the next iteration, in order to avoid getting stuck at local minima. This is a simple practitioner's approach to such a non-convex optimization problem [64].

7.2 Poisson-Gaussian VSTs

First, let us consider an example with $\sigma = 0.02$ (meaning $\mu = 0$ and $\alpha = 1$), and unclipped data. We optimize a fifth degree polynomial and a 3/2 degree rational function (6.4), with the cost functional weight parameters $\lambda_o = 1$, $\lambda_u = 1$, and the constraints on skewness and kurtosis both equalling $\lambda_s = \lambda_k = 0.01$. Figure 7.1(a) presents the obtained optimized VSTs in comparison with the GAT. In particular, we see that the rational VST adheres to the GAT for high intensities, as designed. However, in order to achieve good stabilization for the low intensities, the rational VST exhibits a spike, as is best seen in the zoomed-in portion. On the other hand, the polynomial is naturally unable to produce any such spike. Figure 7.1(b) shows

that the spike of the rational VST indeed leads to excellent stabilization also for the low intensities, whereas the polynomial is unable to even match the performance of the GAT.

For the affine-variance Gaussian approximation of alternative Poisson-Gaussian model (3.16), we look at an example with $a = 0.004$ and $b = 0.02^2$, and the data clipped to $[0, 1]$. Figure 7.2(a) shows the clipped unstabilized standard deviation, and for comparison also the unclipped standard deviation having a square-root form. In Figure 7.2(b), we see again that the rational VST matches the GAT closely, except near the right clipping point. Furthermore, Figure 7.2(c) shows the effect of this behaviour of the rational VST: the corresponding stabilized standard deviation stays relatively close to unity also near the said clipping point, whereas for the GAT, it decreases towards zero much faster. For the polynomial VST, we see similar behaviour as for the rational VST, but with significant oscillations along the whole range.

Note that we have settled on a fifth degree polynomial as a compromise between flexibility and oscillatory behaviour; lower degrees are less flexible, but higher degrees tend to result in more oscillations and numerically unstable behaviour during the optimization. Moreover, finding suitable values for the skew and kurtosis weights λ_s, λ_k is also dependent on whether good stabilization is preferred over good normalization, or vice versa. For a low-degree polynomial, having these constraints restricts its flexibility even more, in which case it is often advantageous to set $\lambda_s = \lambda_k = 0$. For the rational VST, the effect of these weights is less pronounced.

Finally, let us mention that in [12], where we first introduced rational VSTs for the Poisson-Gaussian model (3.11), we optimized 200 VSTs f_σ in the range $\sigma \in [0, 4]$; for larger values of σ , the GAT is already accurate. These VSTs were then successfully used in the variance-stabilization based noise parameter estimation algorithm presented in the paper, providing superior results to what was obtained with the GAT.

7.3 Rice VSTs

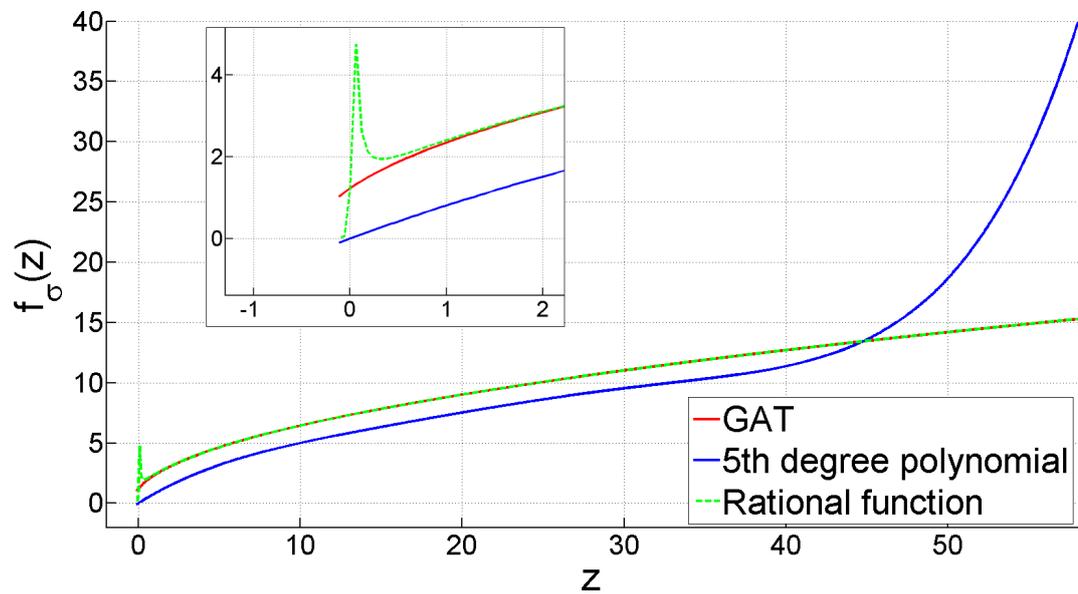
Our first Rice example involves unclipped data with $\sigma = 1$; recall that we can assume $\sigma = 1$ without loss of generality. Now, we optimize a fifth degree polynomial, and both the 3/1 degree rational function (6.10) and the 4/2 degree rational function (6.9). The cost functional weight parameters are the same as in the Poisson-Gaussian examples.

Figure 7.3(a) shows the skewness of the stabilized data for these three optimized VSTs; the rational VSTs are more successful in minimizing the skewness than the polynomial, both asymptotically and in the low-intensity range. Figure 7.3(b) shows the corresponding graphs for kurtosis; while the rational VSTs are again asymptoti-

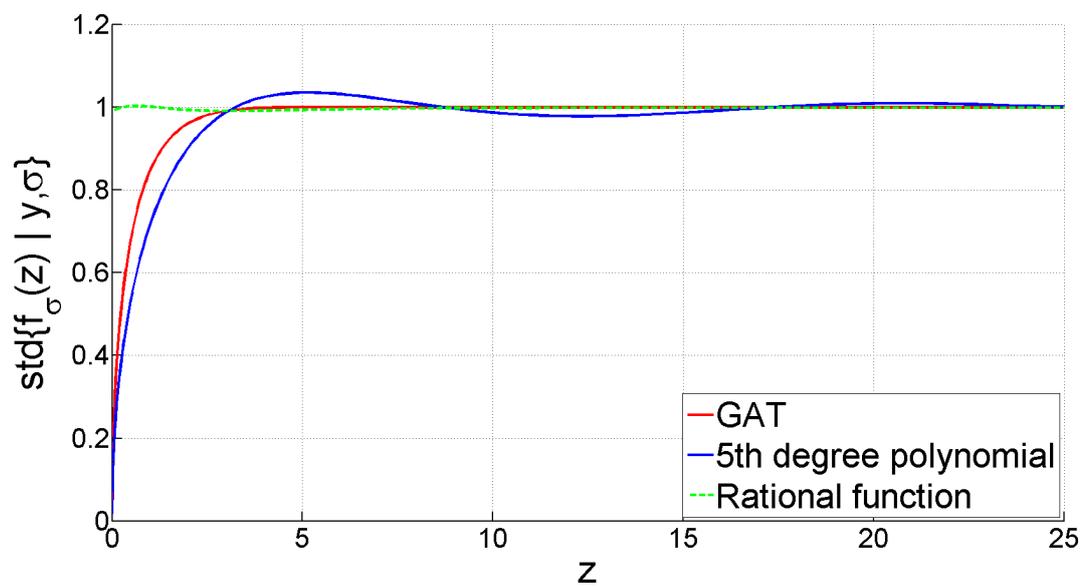
cally more effective, the polynomial is now slightly better at minimizing the kurtosis for low intensities. The asymptotic Rice VST is not included in these figures, as it yields much larger skewness and kurtosis values. Finally, Figure 7.3(c) shows the stabilized standard deviations for all four stabilizers, including the asymptotic Rice VST. Again, for the polynomial VST, there are noticeable oscillations and a lack of asymptotic convergence. On the other hand, both of the rational VSTs outperform the asymptotic VST at the lowest intensities, with the 4/2 degree VST being the most effective; it also converges faster than the 3/1 degree VST.

Our final example considers Rice noise clipped to $[0, 15]$, with $\sigma = 1$. In particular, we demonstrate the effect of the weight parameters λ_s, λ_k , and the effect of the polynomial degree: we optimize a fifth degree polynomial with both $\lambda_s = \lambda_k = 0$ and $\lambda_s = \lambda_k = 0.01$, a seventh degree polynomial with $\lambda_s = \lambda_k = 0$, and the 4/2 degree VST ($\lambda_s = \lambda_k = 0.01$) for comparison.

Figure 7.4(a) shows the optimized VSTs. Further, Figure 7.4(b) visualizes that, at least in this case, disabling the skewness and kurtosis constraints for the polynomials yields better stabilization accuracy. Moreover, the seventh degree polynomial outperforms the fifth degree ones, despite some numerical instability occurring during the optimization. However, the 4/2 degree rational VST once again delivers the best results.



(a)



(b)

Figure 7.1: Unclipped Poisson-Gaussian noise with $\sigma = 0.02$, for the GAT, a 5th degree optimized polynomial VST, and a rational optimized VST. (a) Forward VSTs. (b) Stabilized standard deviations obtained with the VSTs in (a).

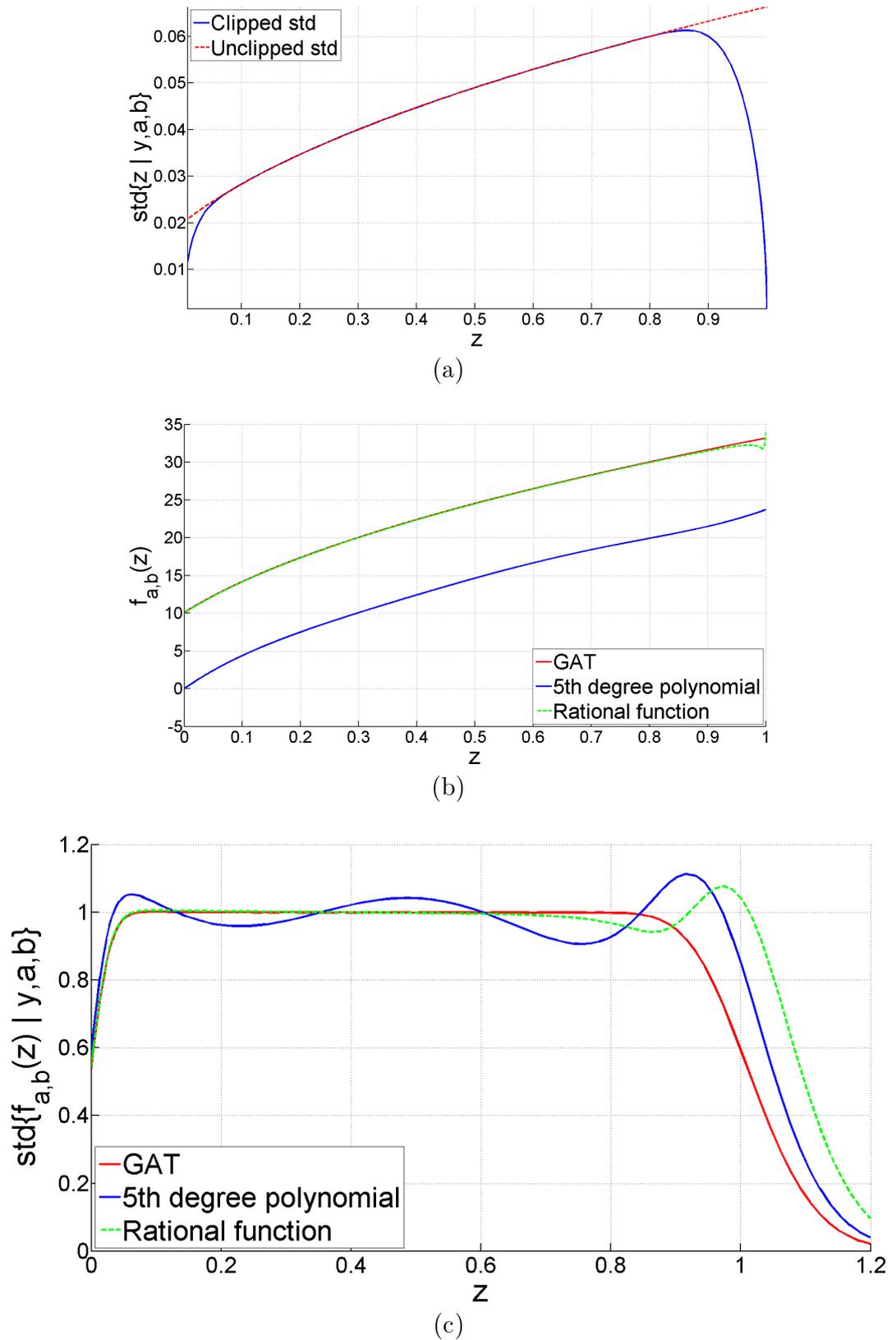


Figure 7.2: Affine-variance Gaussian approximation of Poisson-Gaussian noise with $a = 0.004$ and $b = 0.02^2$, clipped to $[0, 1]$. (a) Clipped and unclipped (unstabilized) standard deviation. (b) Forward VSTs. (c) Stabilized standard deviations obtained with the VSTs in (b).

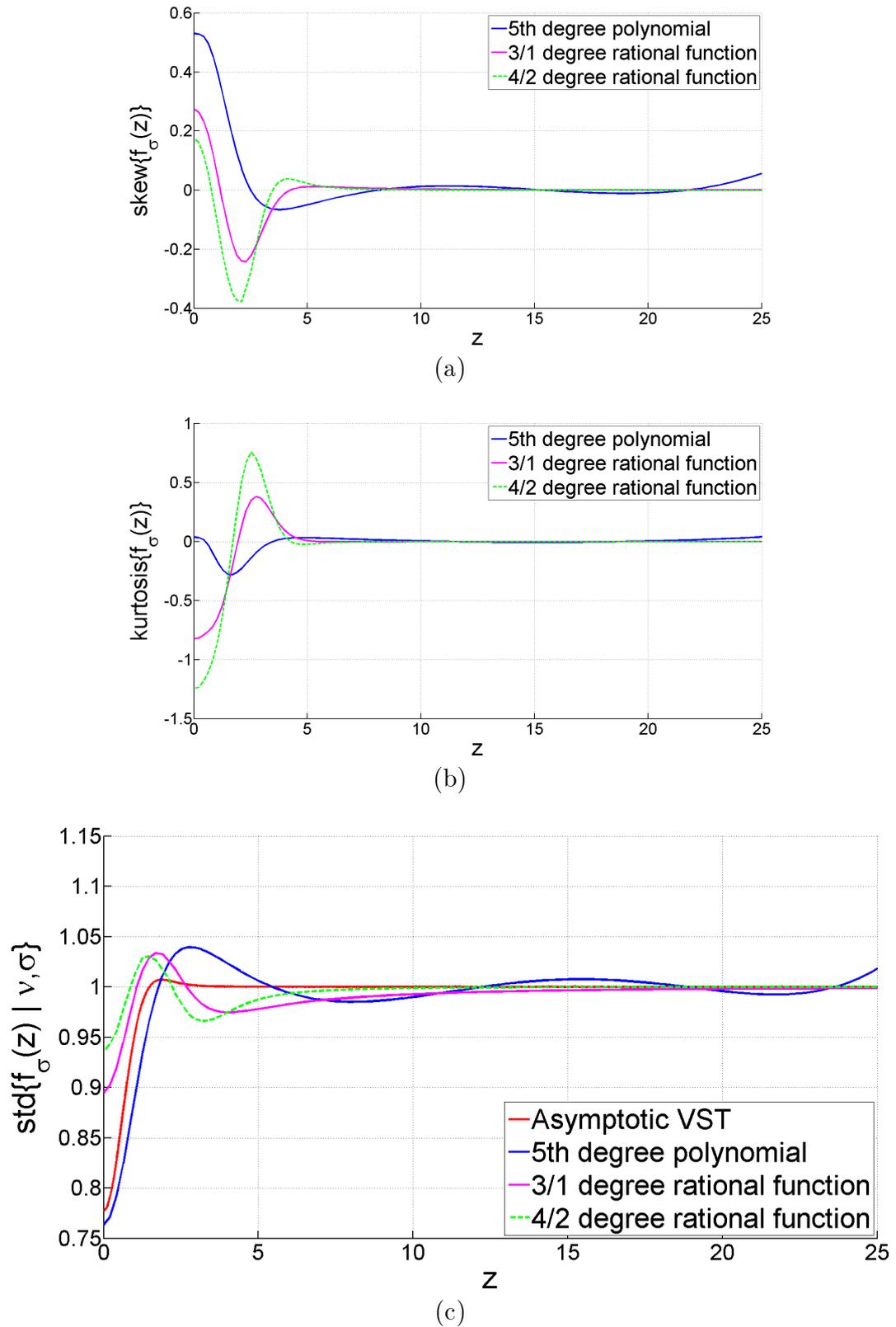


Figure 7.3: Unclipped Rice data with $\sigma = 1$. (a) Skewness of the stabilized data. (b) Kurtosis of the stabilized data. (c) Stabilized standard deviations, including the asymptotic Rice VST.

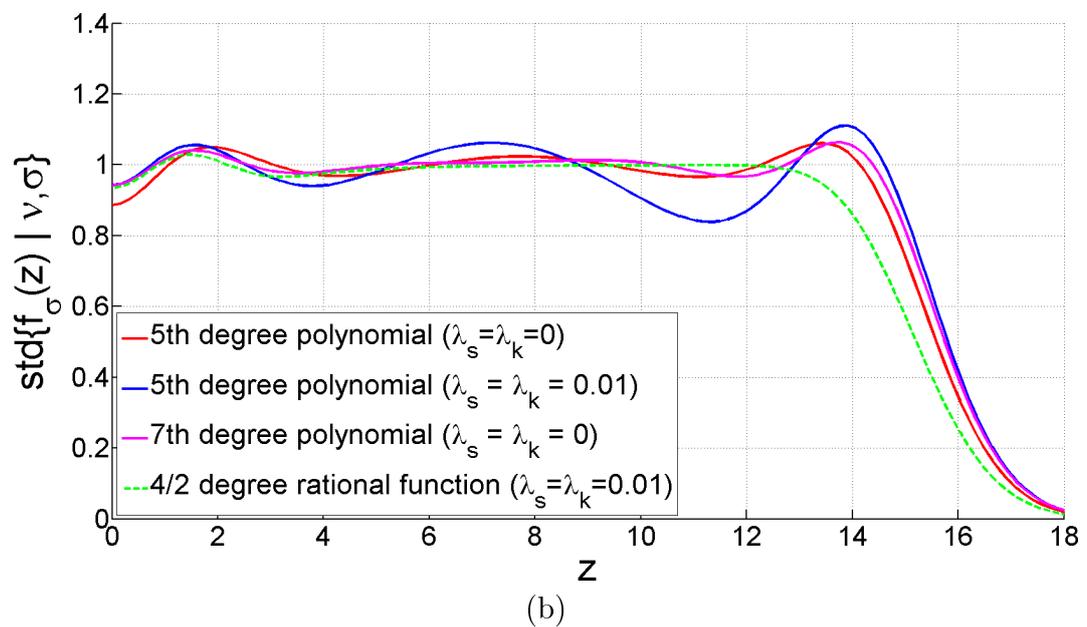
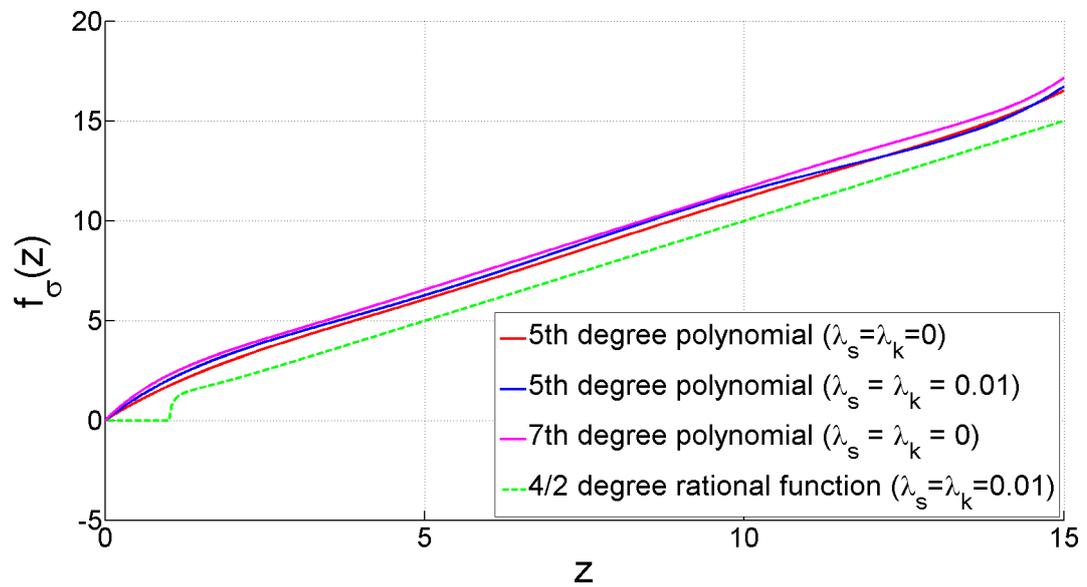


Figure 7.4: Rice noise with $\sigma = 1$, clipped to $[0, 15]$. (a) Forward VSTs. (b) Stabilized standard deviations obtained with the VSTs in (a).

8. CONCLUSIONS

We reviewed some of the theory related to function interpolation and approximation, particularly in relation to polynomials, splines and rational functions. Then we discussed the variance stabilization of Poisson, Poisson-Gaussian, Rice and Rayleigh distributed data, and concluded that traditional parametric variance-stabilizing transformations (VST), which are designed based on asymptotics, provide inadequate stabilization especially for low-intensity data.

Motivated by this deficiency, we established a general framework for optimizing nonmonotonic parametric VSTs, considering the problem as an explicit optimization problem as was done in [11] for nonparametric VSTs. Specifically, we minimize the discrepancy between the stabilized variance and the unitary target variance, while also ensuring the existence of the exact unbiased inverse of the VST. In addition to stabilization, we can affect the amount of normalization by optionally penalizing the skewness and kurtosis of the stabilized data. First, we explored the simple case of polynomials, and then proceeded to the more flexible rational functions, allowing us to set asymptotic constraints, for instance.

We created a modular Matlab toolbox, explicitly implementing the optimization of polynomial and rational VSTs for the Poisson-Gaussian model (3.11), the affine-variance Gaussian approximation of the Poisson-Gaussian model (3.16), and the Rice distribution. However, the framework is extensible to other distributions and other forms of VSTs, as much of the optimization is done independently of them. It is not a fully automatized method, but rather a tool that enables customization and fine-tuning of various parameters based on which properties of the VST are most important for the application at hand.

The experimental results show that rational VSTs consistently provide better stabilization than the asymptotic VSTs. On the other hand, the polynomial VSTs typically produce unwanted oscillations in the stabilized variance, although in some cases they may outperform the asymptotic VSTs, if the polynomial degree and the weights are chosen carefully.

Regarding further research related to the topics discussed in this thesis, obvious directions would be to consider implementing other noise distributions, experimenting with other types of VSTs, and exploring ways to improve the cost functional.

BIBLIOGRAPHY

- [1] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, “Practical Poissonian-Gaussian noise modeling and fitting for single image raw-data”, *IEEE Trans. Image Process.*, vol. 17, no. 10, pp. 1737–1754, Oct. 2008.
- [2] H. Gudbjartsson and S. Patz, “The Rician distribution of noisy MRI data”, *Magn. Reson. Med.*, vol. 34, pp. 910–914, Dec. 1995.
- [3] M. Mäkitalo, *Exact Unbiased Inverse of the Anscombe Transformation and its Poisson-Gaussian Generalization*, Ph.D. Thesis, Tampere University of Technology, Finland, March 2013. <http://urn.fi/URN:ISBN:978-952-15-3039-5>
- [4] F. J. Anscombe, “The transformation of Poisson, binomial and negative-binomial data”, *Biometrika*, vol. 35, no. 3/4, pp. 246–254, Dec. 1948. <http://dx.doi.org/10.1093/biomet/35.3-4.246>
- [5] J. L. Starck, F. Murtagh, and A. Bijaoui, *Image Processing and Data Analysis*, Cambridge University Press, Cambridge, 1998.
- [6] A. Foi, “Noise Estimation and Removal in MR Imaging: the Variance-Stabilization Approach”, *Proc. 2011 IEEE International Symposium on Biomedical Imaging, ISBI 2011*, Chicago, USA, pp. 1809–1814, Apr. 2011.
- [7] M. Mäkitalo and A. Foi, “Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise”, *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 91–103, Jan. 2013. <http://dx.doi.org/10.1109/TIP.2012.2202675>
- [8] J. Salmon, Z. Harmany, C-A. Deledalle, and R. Willett, “Poisson Noise Reduction with Non-local PCA”, *J. Math. Imaging Vis.*, vol. 48, no. 2, pp. 279–294, Feb. 2014.
- [9] R. Giryes and M. Elad, “Sparsity Based Poisson Denoising with Dictionary Learning”, 2014, preprint available at <http://arxiv.org/pdf/1309.4306v2.pdf>.
- [10] A. Foi, “Direct optimization of nonparametric variance-stabilizing transformations”, presented at *8èmes Rencontres de Statistiques Mathématiques, CIRM*, Luminy, Dec. 2008.
- [11] A. Foi, “Optimization of variance-stabilizing transformations”, 2009, preprint available at <http://www.cs.tut.fi/~foi/>.
- [12] M. Mäkitalo and A. Foi, “Noise parameter mismatch in variance stabilization, with an application to Poisson-Gaussian noise estimation”, *IEEE*

- Trans. Image Process.*, vol. 23, no. 12, pp. 5348–5359, Dec. 2014.
<http://dx.doi.org/10.1109/TIP.2014.2363735>
- [13] G. Mastroianni and G. V. Milovanović, *Interpolation Processes: Basic Theory and Applications*, Springer-Verlag, Heidelberg, Berlin, 2008.
- [14] E. Meijering, “A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing”, *Proc. IEEE*, vol. 90, no. 3, pp. 319–342, 2002.
- [15] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed, John Wiley & Sons, New York, 1989.
- [16] P. J. Davis, *Interpolation and Approximation*, Dover Publications, USA, 2014.
- [17] Staff of Research and Education Association, *The Linear Algebra Problem Solver*, Revised Printing, Research and Education Association, New York, 1986.
- [18] V. V. Prasolov, *Problems and Theorems in Linear Algebra*, American Mathematical Society, Rhode Island, 1994.
- [19] J. de Villiers, *Mathematics of Approximation*, Atlantis Press, Paris, 2012.
- [20] D. Kincaid and W. Cheney, *Numerical Analysis*, Brooks/Cole Publishing Company, Pacific Grove, California, 1991.
- [21] J-P. Berrut and L. N. Trefethen, “Barycentric Lagrange Interpolation”, *SIAM Review*, vol. 46, no. 3, pp. 501–517, 2004.
- [22] N. J. Higham, “The numerical stability of barycentric Lagrange interpolation”, *IMA Journal of Numerical Analysis*, vol. 24, no. 4, pp. 547–556, 2004.
- [23] R. F. Churchhouse (Editor), *Ledermann Handbook of Applicable Mathematics. Volume III: Numerical Methods*, John Wiley & Sons, New York, 1981.
- [24] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
- [25] K. Höllig and J. Hörner, *Approximation and Modeling with B-Splines*, SIAM, Philadelphia, 2013.
- [26] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-Spline Techniques*, Springer-Verlag, Heidelberg, Berlin, 2002.
- [27] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes *Computer Graphics: Principles and Practice*, 2nd ed, Addison-Wesley, USA, 1997.

- [28] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed, Springer-Verlag, Heidelberg, Berlin, 1997.
- [29] D. Marsh, *Applied Geometry for Computer Graphics and CAD*, 2nd ed, Springer-Verlag, London, 2005.
- [30] P.L. Chebyshev, “Sur les questions de minima qui se rattachent à la représentation approximative des fonctions”, *Mém. Acad. Sci. Pétersb*, vol. 7, pp. 199–291, 1859.
- [31] K-G. Steffens, G. A. Anastassiou, *The History of Approximation Theory: From Euler to Bernstein*, Birkhäuser, Boston, 2006.
- [32] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed, Springer-Verlag, New York, 1983.
- [33] P.R. Graves-Morris and T.R. Hopkins, “Reliable Rational Interpolation”, *Numerische Mathematik*, vol. 36, pp. 111–128, 1981.
- [34] K.E. Atkinson and W. Han, *Theoretical Numerical Analysis: A Functional Analysis Framework*, 3rd ed, Springer Science+Business Media, New York, 2009.
- [35] W. Gautschi, *Numerical Analysis*, 2nd ed, Birkhäuser, New York, 2012.
- [36] P. Erdős and A.R. Reddy, “Rational Approximation”, *Advances in Mathematics*, vol. 21, pp. 78–109, 1976.
- [37] E.H. Kaufman and G.D. Taylor, “Uniform Approximation by Rational Functions Having Restricted Denominators”, *Journal of Approximation Theory*, vol. 32, pp. 9–26, 1981.
- [38] E.W. Cheney and T.H. Southard, “A survey of methods for rational approximation, with particular reference to a new method based on a formula of Darboux”, *SIAM Review*, vol. 5, no. 3, pp. 219–231, July 1963.
- [39] P.P. Petrushev and V.A. Popov, *Rational Approximation of Real Functions*, Cambridge University Press, Cambridge, 2011.
- [40] NIST/SEMATECH e-Handbook of Statistical Methods, Chapter 4.6.4.2: Rational Function Models. Available: <http://www.itl.nist.gov/div898/handbook/> [Retrieved on April 25, 2015]
- [41] H.C. Tijms, *Understanding Probability: Chance Rules in Everyday Life*, 2nd ed, Cambridge University Press, Cambridge, July 2007.

- [42] H. Lu, X. Li, I-T. Hsiao, and Z. Liang, “Analytical Noise Treatment for Low-Dose CT Projection Data by Penalized Weighted Least-Square Smoothing in the K-L Domain”, *Proc. SPIE, Medical Imaging 2002*, vol. 4682, pp. 146–152, 2002.
- [43] P. Koczyk, P. Wiewiór, and C. Radzewicz, “Photon counting statistics – Undergraduate experiment”, *American Journal of Physics*, vol. 64, no. 3, pp. 240–245, March 1996.
- [44] L. Mandel, “Fluctuations of Photon Beams: The Distribution of the Photo-Electrons”, *Proc. Phys. Soc.*, vol. 74, no. 3, pp. 233–243, 1959.
- [45] S. O. Rice, “Mathematical Analysis of Random Noise”, *Bell System Technical Journal*, vol. 23, pp. 282–332, 1944.
- [46] S. O. Rice, “Mathematical Analysis of Random Noise (Part III)”, *Bell System Technical Journal*, vol. 24, pp. 46–156, 1945.
- [47] G. N. Watson, *A Treatise on the Theory of Bessel Functions*, 2nd ed, Cambridge University Press, Cambridge, 1944.
- [48] A. N. Celik, “A statistical analysis of wind power density based on the Weibull and Rayleigh models at the southern region of Turkey”, *Renewable Energy*, vol. 29, no. 4, pp. 593–604, Apr. 2004.
- [49] M. Mäkitalo, A. Foi, D. Fevraleev, and V. Lukin, “Denoising of single-look SAR images based on variance stabilization and nonlocal filters”, *Proc. Int. Conf. Math. Meth. Electromagn. Th., MMET 2010*, Kiev, Ukraine, pp. 1–4, Sept. 2010. <http://dx.doi.org/10.1109/MMET.2010.5611418>
- [50] H. Suzuki, “A Statistical Model for Urban Radio Propagation”, *IEEE Trans. Commun.*, vol. 25, no. 7, pp. 673–680, July 1977.
- [51] A. Foi, “Clipped noisy images: heteroskedastic modeling and practical denoising”, *Signal Processing*, vol. 89, no. 12, pp. 2609–2629, Dec. 2009. <http://dx.doi.org/10.1016/j.sigpro.2009.04.035>
- [52] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3D transform-domain collaborative filtering”, *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
- [53] J. Mairal, G. Sapiro, and M. Elad, “Learning multiscale sparse representations for image and video restoration”, *Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

- [54] L. H. C. Tippett, “Statistical Methods in Textile Research, Part 2 — Uses of the Binomial and Poisson Distributions in Analysis of Variance”, *Journal of the Textile Institute Transactions*, vol. 26, no. 1, pp. T13–T50, 1935.
- [55] J. H. Curtiss, “On transformations used in the analysis of variance”, *The Annals of Mathematical Statistics*, vol. 14, no. 2, pp. 107–122, June 1943.
- [56] B. Efron, “Transformation theory: How normal is a family of distributions?”, *The Annals of Statistics*, vol. 10, no. 2, pp. 323–339, 1982.
- [57] R. Tibshirani, “Estimating Transformations for Regression Via Additivity and Variance Stabilization”, *Journal of the American Statistical Association*, vol. 83, no. 402, pp. 394–405, June 1988.
- [58] M. S. Bartlett, “The Square Root Transformation in Analysis of Variance”, *Supplement to the Journal of the Royal Statistical Society*, vol. 3, no. 1, pp. 68–78, 1936.
- [59] M. Freeman and J. Tukey, “Transformations related to the angular and the square root”, *The Annals of Mathematical Statistics*, vol. 21, no. 4, pp. 607–611, 1950.
- [60] P. R. Prucnal and E. L. Goldstein, “Exact variance-stabilizing transformations for image-signal-dependent Rayleigh and other Weibull noise sources”, *Applied Optics*, vol. 26, no. 6, pp. 1038–1041, March 1987.
- [61] M. Mäkitalo and A. Foi, “Optimal inversion of the Anscombe transformation in low-count Poisson image denoising”, *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 99–109, Jan. 2011. <http://dx.doi.org/10.1109/TIP.2010.2056693>
- [62] M. Mäkitalo and A. Foi, “A closed-form approximation of the exact unbiased inverse of the Anscombe variance-stabilizing transformation”, *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2697–2698, Sept. 2011. <http://dx.doi.org/10.1109/TIP.2011.2121085>
- [63] S. K. Bar-Lev and P. Enis, “On the construction of classes of variance stabilizing transformations”, *Statistics & Probability Letters*, vol. 10, pp. 95–100, July 1990.
- [64] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, MPS-SIAM Series on Optimization, vol. 8, MPS-SIAM, Philadelphia, 2009.