Finding Algebraic Mathematical Models from Experimental Data with Artificial Intelligence

Finding Algebraic Mathematical Models from Experimental Data with Artificial Intelligence

By

Angel Fernando Kuri-Morales

Cambridge
Scholars
Publishing

Finding Algebraic Mathematical Models from Experimental Data with Artificial Intelligence

By Angel Fernando Kuri-Morales

This book first published 2026

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-1-0364-5995-6
ISBN (Ebook): 978-1-0364-5996-3

This work is dedicated to my daughter Leticia and my son Angel.
And to their sons and daughters Antonio, Alessandro, Santiago, Ana Paula and Mariam.
They are the five leaves of the magic clover.
May one day the diurnal sun and moon may shine together again.

TABLE OF CONTENTS

CHAPTER 1

FINDING ALGEBRAIC MATHEMATICAL MODELS FROM EXPERIMENTAL DATA WITH AI

1. Introduction

This book is about modeling. In particular it is about how to obtain closed models from numerical data by taking advantage of the techniques of artificial intelligence (AI).

Most of us appeal to models in many aspects of our every day lives. Models allow us to make intelligent decisions when facing new unknown situations once we have determined the variables which will allow us to predict the behavior of a particular instance for a given problem. Any real life problem, of course, offers us a myriad of alternatives when we decide what is important to face such problem. Whether we are aware of it or not, when we use our knowledge, we have already selected what is important about a particular situation and what is not. For instance, when planning to take a trip we select its destination and the date in which we plan to do it. We would like to predict the cost of the trip. In this particular case the independent variables are the destination and the date, while the cost of the trip is a dependent variable. It depends on the selection of the destination and the date. But we do not question ourselves on whether the number of days will change between today's date and the selected date of departure. We know, from experience, that every day has a 24 hour duration regardless of the city in which we reside or the month or the year of our planned trip.

In this very simple example, we have modeled the cost of the trip as a function of the variables of interest. We know which rules govern the way in which destination and date bear on the cost of the trip. We pay no attention to whether the allotted time will inexplicably change because we assume that time periods are usually constant. But we certainly will choose the values of the independent variables which will minimize the value of the cost.

In most cases, actually, the rules which govern the behavior of the dependent variable given the values of the independent variables are unknown. However, if we are able to reliably quantify the values of both the dependent and independent variables, we may infer a model out of these values. To be able to extricate models from data in a generalized sense the involved tools are somehow sophisticated. Nowadays some of the said tools depend on state-of-the art techniques related to artificial intelligence. In fact, modern science may be thought of as the art of discovering the models which underlie the reasons why a dependent variable behaves as it does given that the independent variables assume the values they do.

In this book only metric variables are considered[1]. The methodology which we will describe may be extended to non-metric (*categorical*) variables if a proper encoding system is applied. In such case non-metric variables may be mapped onto their metric counterparts and this limitation is superseded. The interested reader will find several such encoding schemes analyzed in [1].

The formal meaning for the term 'model' in scientific disciplines differs from the everyday usage of the word. This is important in understanding everyday experiences with models. In fact, 'models' can be more than just physical representations of objects. Interestingly, 'models' can also be used to test ideas and processes in ways that may be impossible to do in the real world. This notion might be reinforced by the everyday use of such terms as 'computer modeling' and experimental results of computer game simulations. A more advanced view, rarely encountered, in general, is that testing 'models' (the act of testing for 'fit') can lead to their redesign to give improved predictions. Few people are aware of the importance placed on testing and refining models to build better understanding of the processes they seek to explain [2].

Models are typically used when it is either impossible or impractical to create experimental conditions in which we can directly measure outcomes. Direct measurement of outcomes under controlled conditions will always be more reliable than modeled estimates of outcomes. But oftentimes an accurate model may yield new insights to the object being modeled. If we were able to establish a methodology leading to the obtention of an explanatory model from a set of experimental data we would, in principle, be able to explore the relations therein exhibited. Furthermore, we would, in principle, also be able to understand the origin of the patterns expressed by our model.

[1] In mathematics a metric space is a set together with a notion of distance between its elements, usually called points. The distance is measured by a function called a metric or distance function.

A model is a task-driven, purposeful simplification and abstraction of a perception of reality, shaped by the appropriate constraints [3]. It is task-driven because a model is captured with a certain question or task in mind. Simplifications leave all the known and observed entities and their relation out of what is not important for the task. Abstraction aggregates information that is important but not needed in the same detail as the object of interest. Simplification and abstraction are done purposefully. However, they are done based on a perception of reality. This perception is already a model in itself, as it comes with a physical constraint. There are also cognitive constraints that limit what we are able to explain with our current theories. This model comprises the concepts, their behavior, and their relations informally and is often referred to as a conceptual model. In order to execute the model, it is frequently implemented as a computer simulation. This requires more choices, such as numerical approximations or the use of heuristics. Despite all these epistemological and computational constraints, simulation via modeling has been recognized as the third pillar of scientific methods: theory building, simulating, and experimenting [4].

The individuals of the intellectually more developed species (such as Homo sapiens) are able to design a complex set of actions leading to the final purported goal. For example, to prove the existence of the Higgs field, man had to model it. He had to show, first, that a boson [5] arising from such hypothesized field existed. To do this a very large and complex proton accelerator machine (LHC) had to be designed and built. This machine was able to accelerate the protons to almost the speed of light. The design specifications of the LHC relied on a very large number of previously verified models. The fact that the speed of light may never be reached is, itself, a consequence of a previous model (Einstein's Theory of Special Relativity) which has been exhaustively tested and confirmed. The LHC was built by the European Organization for Nuclear Research (CERN) between 1998 and 2008 in collaboration with over 10,000 scientists and hundreds of universities and laboratories, as well as more than 100 countries [6]. The said collaboration, as well, is an example of a very intricate collection of interrelated models. The World Wide Web (WWW) was originally conceived, for example, to allow for the scientists in the collaboration to efficiently communicate between themselves [7]. To achieve the desired acceleration of the protons in LHC a method to focus a set of batches of protons circulating in the rings, specifically designed for that purpose, was envisioned by Simon van der Meer [8]. When these highly energetic protons collided they produced a large variety of subatomic particles. Among them there were copious neutrinos and antineutrinos which accounted for the "missing" mass after the protons collided in the LHC. The proof of the existence of such almost massless particles was the result of a convoluted chain of reasoning and the associated models and experiments [9]. If these normally undetectable particles had not been accounted for, the dimensions of the LHC's main tunnel would have been impossible to determine. And, thus, the Higgs field would have been impossible to find. A large number of models was involved and it was the methodological tool behind this unprecedented discovery. It is not an overstatement to say that the evolution of humankind has rested behind countless chains of models leading to ever more sophisticated ones in an ascending spiral.

To arrive at a valid model of any given system the typical path has been to identify the relative importance of such system, observe its characteristics and try to infer the patterns present in it and, ultimately, the underlying reasons for those patterns to appear. Tacit in this methodology is the belief that, once such patterns have been disclosed, the behavior of the system will follow the same rules which led to their discovery.

In the more recent years humans have been able to measure and quantify a very large set of phenomena. It is accepted that the amount of data gathered from the scientific and social endeavors grows exponentially. Over the next years (up to 2025) global data creation is projected to grow to more than 180 zettabytes[2] [4]. Therefore, it is clearly obvious that being able to extract models from the data is a potential subject of great practical importance.

Our main interest is to explore the possibility of designing mathematical methods which will allow us to find the models behind the phenomenon under study. In modern parlance, we attempt to find a meta-model: a model to find models.

1.1. Expert Systems

Originally, automated models were sought by consulting experts in the field from which data was obtained. Ideally, these experts were able to deliver the rules which allowed a non-expert to achieve results similar to those which an expert would have attained, but without the need to have the expert's knowledge. These models were known as "Expert Systems". Once the set of rules was identified, basic facts were fed to the model. Following the steps of the expert's rules even an unsophisticated individual would be able to replicate the logic behind the model and achieve the desired conclusions given a basic set of data items.

A major inconvenience of this approach is that the purported expert has to know the rules which are being applied in order to get the conclusions. Unfortunately there are many cases in which the expert is unable to deliver the rules. It is simply not known how the diagnostics are arrived at. Frequently the expert is cited as saying that his opinion is derived by "intuition".

[2] A unit of data equal to one sextillion (10^{21}) or, strictly, 2^{70} bytes.

Another, and perhaps more serious, inconvenience is that to use the rules, these have to be applied in a pre-specified order. Once rule 1 is applied it leads to a conclusion where rule 2 may be applied; once rule 2 is applied rule 3 may be applied and so on. But what if, say, rule 3 depends (even partially) on the conclusions of, say, rule 7? The prescribed application of the rules must ensure that this kind of backward logical loops does not exist anywhere in the deductive model.

Finally, even if rules are correctly identified and logical loops are avoided, a more subtle inconvenience is that there is no way to ascertain the consistency of the system's rules. But, unfortunately, there is no way to prove that there are no rules which may contradict each other.

1.2. Deductive Reasoning

Deductive reasoning has been explored since, at least, the ancient Greek philosophers. To start a valid reasoning sequence of logical steps leading to a valid conclusion, one must accept some a priori valid truths called "axioms". Aristotle famously derived a "formal" system, leading to the so-called syllogisms. These are a form of reasoning in which a conclusion is drawn from two given or assumed premises, each of which shares a term with the conclusion, and shares a common term not present in the conclusion. It is formal because as long as the structure of a syllogism is preserved, the conclusions are valid. It does not base its efficacy in the meaning of the terms. As long as the premises are true AND the structure of the syllogism is kept, the conclusion will be also true. It is the epitome of deductive reasoning. Aristotle discusses 15 valid forms of categorical syllogisms. Names were assigned to each by medieval logicians. For example, "Barbara" is a syllogism of the first so-called "figure" (where the "A" stands for universal affirmative). It does have a AAA structure. An example of Barbara is: All men are mortal (Major premise), John is a man (Minor Premise) therefore (conclusion) John is mortal. We *assume* that "All men are mortal" and "John is a man" are both true, hence "AA" logically leads to "John is mortal" (also "A").

A set of axioms is consistent if there is no statement such that both the statement and its negation are provable from the axioms and inconsistent otherwise. This matter has been extensively analyzed by logicians and, in this regard, Kurt Gödel proved the so-called Second Incompleteness Theorem: "For any consistent system F within which a certain amount of elementary arithmetic can be carried out, the consistency of F cannot be proved in F itself." [10]

For these reasons Expert Systems, although still popular in some disciplines, have been superseded by a more recent approach which does not look for the rules behind a model but, rather, relies on the discovery of the relations which lead to a different kind of models where the aim is to find a function which expresses the relation between a variable which depends on the values of other variables; even if the rules implicit in so doing are not known.

1.3. Learning from the Data

Because of the importance of being able to learn from the data (i.e. being able to find an adequate model stemming from a set of data), many methods have been proposed which aim at automating the process of learning from the data. This field has come to be called "Machine Learning" (or ML) and is part of artificial intelligence. The associated algorithms are sometimes called "Learning Machines" (or LM). Perhaps the most ubiquitous LM is embodied in what are commonly known as "Artificial Neural Network" models or NN [11]. Close to 12,000 peer reviewed scientific papers on NNs are published every year [12].

We shall consider only those NNs where, given a set of input values, there is a unique corresponding output value. This paradigm is called supervised learning. It is called supervised because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers and the algorithm iteratively makes predictions on the training data and is corrected by the teacher.

A NN receives as an input a set of values of the independent variables and the corresponding value of a dependent variable. The set of independent variables is denoted by $\vec{x}$; the dependent variable is denoted by *y*. Hence, $\vec{x}$ consists of a set of values for each of the *attributes* of the model. *y* is the value dependent of the attributes in $\vec{x}$. One example with which the NN is fed is called a training couple [denoted by $(\vec{x}, y)$]. A training couple is called a *tuple*. Therefore, for our purposes, a database is, simply, a set of training tuples.

NNs have been successfully used in innumerable occasions with great success. When faced by a large set of $\vec{x}$ they are able to deliver the corresponding *y* accurately. In other words, an adequately trained NN constitutes a model of the phenomenon represented by the $(\vec{x}, y)$ tuples. This is a remarkable achievement because it allows us to find models for which the underlying relations between the independent variables are unknown. This is why NNs are considered to be "black box" models. That is, even though an NN is able to deliver the right answers to a given problem given a set of input values, the way it does so (not the methodology to train a NN) remains unknown.

One of the main purposes of this book is to propose a methodology to remove the “black box” characteristic from the model in case.

The problem of finding a synthetic expression from an experimental set of data for a variable of interest given a matching set of independent variables has repeatedly received attention. It may be called multivariate regression or supervised training (ST) depending on the approach considered. As ST it has given rise to a wide sub-area within the realm of artificial neural networks. Some approaches include multilayer perceptron networks [13] [14], radial basis function networks [15] [16] and support vector machines [17] [18]. A main concern has been, as stated, the fact that the architecture of the model yields little information of the relations between the variables of the system. On the other hand, if seen as a multivariate regression problem where, conceivably, such relation is explicit, the following issues arise: a) How to determine the form of the model [19], b) How to calculate the free parameters of such model when one is selected [20] and, c) How to handle the complexity of its equations and the numerical instability they frequently convey [21]. The last issue has to do with the fact that a closed model of a system of multiple variables frequently leads to complex expressions whose determination depends on solving large sets of simultaneous linear or non-linear equations. These sets of equations are prone to result in ill-conditioned matrices [22]. In this book the form of the model is left for the method to determine appealing to another tool of AI: evolutionary computation. The use of NNs has opened a way out of the ill-conditioned matrices. NNs represent the embodiment of an alternative approach which replaces one complex expression by sets of simpler ones. The free parameters and architecture of the NN, when adequately determined, yield the desired results [23]. However, the explicit relations between the independent variables, as already remarked, remain unknown. We explore an alternative which allows us to keep the closed nature of an algebraic model (along with its inherent explanatory properties) while avoiding the pitfalls of numerical instability and leaving the task of determining the mathematical model that is best suited for the problem at hand to the method itself.

It is very important to remark that, as opposed to the statistical analysis technique, (which aims a model function to match a data set, where the goal is to find the best model parameters so that the model can be adapted to the data) here we are not looking for a way to best fit a predetermined model. Typically, curve fitting examines the relationship between one or more independent variables and a dependent variable, with the goal of defining a “best fit” model of the relationship. It is the process of constructing a mathematical function, which has the best fit to a series of data points. The important caveat here is that a function to which we want to fit our data has to be set a priori. The parameters that define the model have to be set in advance. Rather, our method finds the best model without assuming a particular case predetermined form.

1.4. Dictum

Therefore, the goal of the methodology behind polynomial multivariate approximation modeling may be simply stated as per the following dictum D:

“Find the coefficients of the linear combination of a set of *m* monomials

$$X_i = \Pi_{j=1}^{n} v_j^{k_j} \text{ where } \sum_{j=1}^{n} k_j \in L(i)$$

such that the approximation error is minimized”.

To understand D we must clearly define and analyze its meaning. This task turned into the book you have in your hands. It is divided in 10 chapters. Chapters 2, 3, 4, 5, 6 and 7 are devoted to explain, analyze and justify D´s validity and generality. Chapters 8, 9 and 10 were designed to concretely illustrate the application of the model with several cases of study.

1.5. Contents

In chapter 2, we analyze the way in which data is stored in a digital computer. The Stone–Weierstrass Theorem [24] (SWAT) implies continuity in the functions to be approximated. We show that, in practice, because of the finiteness of data representation in a digital computer, strict continuity is impossible.

1.5.1. Collocation and Approximation

We discuss the difference between collocation and approximation. We show that polynomial collocation models are prone to Runge’s phenomenon [38] and approach collocation models by using natural splines[25], which are impervious to this phenomenon. We also discuss the possibility of interpolating discontinuous data to enhance it and get an adequate approximately continuous data via natural splines. Next we introduce approximation models.

To be able to approximate (rather than collocate) the data, we must select a measure to quantify the errors inherent in the approximation models. We analyze two possible norms: L_2 and L_∞. Both of these norms are particular cases of the Minkowsky family of norms $\| \varepsilon \|_p = \left(|\varepsilon_1|^p + |\varepsilon_2|^p + \ldots + |\varepsilon_n|^p \right)^{1/p}$. We analyze the L_2 univariate approximation case and show that a practical solution to minimize the L_2 norm is to use Chebyshev Polynomials, which also simultaneously approximately minimize the L_∞ norm. Then we discuss the way to strictly minimize the L_∞ norm. We introduce the Ascent Algorithm (or Exchange Algorithm) and discuss Remez Algorithm which is effective to minimize the L_∞ norm for polynomials in the univariate case. We finally discuss the case where a possible family of univariate models is tested under L_2 to find a suitable form and conclude that this approach may not be adopted when more than one independent variable is involved. There are simply too many alternative possible models.

1.5.2. Multivariate Approximation

Chapter 3 is devoted to analyzing the way multivariate approximation may be approached. Using L_2 to find a multivariate approximant, though possible, is infeasible in practice because it requires the knowledge of vast amounts of tuples (frequently referred to as the "curse of dimensionality") which have to comply with knowledge of the values of the tuples in a grid (a predefined set of values in an *n*-dimensional space). L_∞, on the other hand, does not exhibit any of the two mentioned impediments. It may be used, in practice, by implementing the so-called "Fast Ascent Algorithm"[28] or FAA. We discuss FAA´s implementation.

1.5.3. Multivariate Approximation via the Universal Approximation Theorem

In chapter 4 we show that it is possible to work backwards from the Universal Approximation Theorem (UAT) [26] in the sense that we may derive an algebraic expression starting from a NN architectural approach. We know that the logistic function $1/(1+e^{-x})$ may be used as the nonlinearity in the UAT and that an accurate approximation of the *logistic*, adequate for algebraic manipulation, may be gotten from a Chebyshev polynomial basis [27]. We generalize the UAT by using the explicit algebraic approximation of the *logistic* and calculate the number of elements in the explicit polynomial representation for an arbitrary function. The resulting approximation gives rise to the AUAT (Algebraic Universal Approximation Theorem) which, in turn, leads us to an algebraic model applicable to an arbitrary set of tuples. Accordingly, the experimental model (or approximant) is defined to have the form:

$$Y = c_1X_1+c_2X_2+ \ldots +c_mX_m \tag{1.1}$$

Where m is the number of desired terms and X_i is a monomial which denotes a product of the powers of the n independent variables each raised to a maximum positive degree d. Thus:

$$X_i = \Pi_{j=1}^{n} v_j^{k_j}; \quad 0 \le k_j \le d \tag{1.2}$$

Furthermore, a significant result is that the powers of the monomials in (1.2) (with the exception of a constant value) ought to be odd and satisfy $\sum_{j=1}^{n} k_j \in L(i)$ where L = {0,1, 3, 5, 7, 9, 11, 15, 21, 25, 27, 33, 35, 45, 49, 55, 63, 77, 81, 99,121}. Hence, from the AUAT we are able to bound the maximum number of terms, as well as the highest degrees allowed in any such term. The goal of polynomial multivariate approximation modeling may be finally be stated as what we called "dictum D".

This is a powerful conclusion but the AUAT, unfortunately, is not constructive. To get a generalized modeling tool from it we further need to solve the next problems:

a) Determine how to get a polynomial approximant from an arbitrary set of tuples.
b) Determine the cardinality of m in (1.1).
c) Find a way to comply with X_i in (1.2).

1.5.4. Determination of the Number of Terms of the Multivariate Approximation

In chapter 5 we describe how the best practical number of terms (m) in the approximant may be estimated from a previously trained NN which we denote by NN_T. We arrive at the way to approximately determine m, accounting for problem (b), as follows.

We succinctly describe a method to determine the proper number of terms in our approximant by studying the behavior of the model on a large number of problems. The first step was to set lower and upper practical limits on m. We reasoned that we were not interested in very small values for m. Very low values are hardly prone to yield good models. Therefore, we set a lower value of $m=3$. On the other side of the spectrum we decided to focus on $m \le 13$. Higher values are seldom of practical interest since large sets of coefficients are cumbersome and difficult to analyze.

We collected 46 datasets from the University of California Machine Learning dataset repository [30] and the Knowledge Extraction Evolutionary Learning dataset repository [31]. To begin with, 32 of these datasets were chosen and were solved using our methodology for the model. The remaining 14 data sets were used to validate the learning process. For every $m \in [3,13]$ a polynomial satisfying dictum D was found and the number of terms corresponding to the best fit was recorded. A total of 506 (11 x 46) polynomials, therefore, was calculated. From it the lower bound of the number of terms was determined. The best values of *m* were then used to train a Neural Network (NN_T). Now, for any new database (call it ND), we may determine a good initial choice for *m* from NN_T by feeding it with the parameters corresponding to the characteristics of ND. One of these is the amount of information contained in ND.

1.5.4.1 Quantifying the Information in the Data

It is frequently overlooked that different databases of similar size do not necessarily convey the same amount of information. For example, given ND1 consisting of *T* equal tuples (which holds very little information) and ND2 consisting of *T* random tuples (holding a much larger amount of information) a direct comparison between ND1 and ND2 would be inadequate. To avoid this pitfall, all the databases used to train NN_T were homogenized in the sense that the information in all the 46 databases yielding NN_T correspond to similar amounts of information rather than similar sizes. Therefore, it is imperative that the information of ND be calculated before using NN_T. Hence, information theory, both statistical and algorithmic, is discussed. We show that statistical information is not suitable for our purposes while, on the other hand, algorithmic information (or Kolmogorov´s complexity) is uncomputable [29]. A method which allows us to closely approximate algorithmic information is discussed.

1.5.5. Genetic Algorithms

In chapter 6 we discuss a method which allows us to solve problem (c) by selecting only the most adequate monomials via the use of a genetic algorithm (GA) [32] [33]. GAs are very interesting tools in their own right and there are many variants of them. There is an unlimited number of ways in which crossover and mutation[3] may be attempted. The following 4 structurally different GAs: TGA (eliTist GA), CHC[4] (CHC GA), EGA (Eclectic GA), SGA (Statistical GA) plus an iterative algorithm RMH (Random Mutation Hill Climber) were selected to determine the best choice to explore the models complying with dictum D. These algorithms are described and selected flow diagrams are presented. The 5 algorithms were statistically tested to find the best one. The statistical test consisted of presenting each of the algorithms with a large set of problems. The test only stopped when the distribution of the minimum errors became normal rather than defining a set of arbitrarily selected problems. From the central limit theorem we know that if sufficiently large samples from a population are considered, the samples' means will be normally distributed, even if the population is, itself, not normally distributed. A basic problem, therefore, is how to determine if the samples are normal. The correct way to determine normality of a data set is, therefore, also discussed. In this case, on the order to 70,000 problems were solved to ascertain normality of the distributions of the minimum errors for all and each of the considered algorithms.

We found that EGA's performance was the best in:

1) Unconstrained Functions
2) Constrained Functions
3) Order of Schemata for Unconstrained Functions
4) Order of Schemata for Constrained Functions
5) Deceptive Functions

Second best in

6) Royal Roads Functions

The results of the 5 algorithms´ comparison are summarized in Fig. F1.

Algorithm	Average Minimum	Relative Efficiency
EGA	0.0635	100.00%
SGA	0.1260	50.43%
RMH	0.1491	42.60%
CHC	0.1501	42.32%
TGA	0.2272	27.96%

Figure F1. The Best Genetic Algorithm

[3] For which see chapter 6.
[4] Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm.

From the AUAT and having solved problems (a) (The determination of how to get a polynomial approximant from an arbitrary set of tuples via the Fast Ascent Algorithm), (b) [(How to determine the cardinality of m in (1) via the neural network NN_T] and (c) [(How to comply with X_i in (1.2) via EGA] we are in the position of finding an algebraic model for, virtually, any experimental numerical data set.

1.5.6. Machine Learning with Genetic Multivariate Models

In Chapter 7 we describe the implementation of a comprehensive program which allows the reader to benefit from all the concepts of the preceding chapters. This chapter is included to provide a clear understanding of a program called GMP5 which was explicitly implemented to allow for algebraic models ML.

1.5.7. Cases of Study

In chapters 8, 9 and 10 we apply our method to the following three cases of study:

Chapter 8. The problem of adequately classifying three cultivars (classes) from 13 attributes identified by the experts.

Chapter 9. The problem of determining the relationship between 4 selected physical independent attributes: Element, Mass, Temperature, Volume and a dependent attribute: Energy.

Chapter 10. The problem of, from a set of selected *tickers*[5] (which in this case are the independent variables) determine the future value of another ticker (which becomes the dependent variable), as a function of the historical behavior of the set of independent tickers for a predetermined time lapse.

1.5.7.1. Classification of Wines from Experimental Data

The model for chapter 8 yields the mathematical relation between the 13 attributes selected by the experts and the corresponding classes [34]. The model learns the relation between the classes and a subset of the original sample (the so-called training set). It generalizes the results by testing an unknown subset of the sample (the so-called test set). The model is able to determine the class of each of the elements of the test set with 98% accuracy. Remarkably, and perhaps more importantly, the explicit mathematical expression of the model shows that several of the attributes determined by the experts have no bearing on the correct classification of the wines in the class.

1.5.7.2. Analysis of Sensitivity for Physical Phenomena (Derivation of $e=mc^2$ from Experimental Data)

In chapter 9 a physical phenomenon is investigated. We look for a model exhibiting the relation between Element, Mass, Temperature, Volume and Energy of an arbitrary body. The attribute "Energy" is selected as the dependent variable, while Element, Mass, Temperature and Volume are the independent physical variables. The model becomes

$$Energy = F(Element, Mass, Temperature, Volume) \quad .$$

As a result of modeling the last equation from the given data, the model exhibits the fact that the coefficients of all unrelated independent variables in it are close to zero. This means that in those cases where a variable's coefficients are negligible they have no influence on the dependent variable. The coefficient of the variable "Mass" is the only one significantly different from zero. Its value is approximately equal to 9×10^{16}. The reader will most likely recognize that $9 \times 10^{16} = (3 \times 10^8)^2$ = the speed of light in vacuum (in m/s) squared.

Hence, the model is capable of disregarding the variables "Element", "Temperature" and "Volume" by assigning values close to zero to their associated coefficients while it correctly retains "Mass" by assigning the value "c^2" to its coefficient. That is, the model is able to determine which variables are relevant from a data set where, in principle, nothing is known about them except their numerical values. Therefore, the relation that the model finds between energy (e) and mass (m) is simply "$e = mc^2$".

Of course, whereas the model identifies the relevant variables and the correct relation between "Mass" and "Energy" it cannot tell us why this is so. This was left for Einstein to explain [35].

1.5.7.3. Forecasting Stock Values in the American Trade Market from Experimental Data

In chapter 10 we explore the possibility of forecasting the values of the American stock market. This, in principle, is very difficult, if not downright impossible. A lucid illustration of this fact is clearly treated in the video "The 4 things you need to be an expert" [36]. In it Derek Muller, the author of the video, exemplifies this fact with a bet by Warren

[5] A ticker is a symbol, a unique combination of letters and numbers that represent a particular stock or security listed on an exchange. The symbol is used to refer to a specific stock, particularly during trading. Trades are executed based on a company's ticker symbol, which is recorded in the exchange's trading system.

Buffet vs. Ted Seides. Seides used hedge funds while Buffet used a passive index fund (S&P 500). For the record, Seides lost the bet.

The message is that stock market investing is not prone to the application of simple expert rules. From [37] we quote "*In finance, technical analysis is an analysis methodology for analyzing and forecasting the direction of prices through the study of past market data, primarily price and volume. As a type of active management, it stands in contradiction to much of modern portfolio theory. The efficacy of technical analysis is disputed by the efficient-market hypothesis, which states that stock market prices are essentially unpredictable, and research on whether technical analysis offers any benefit has produced mixed results. It is distinguished from fundamental analysis, which considers a company's financial statements, health, and the overall state of the market and economy.*"

Nonetheless, after recent exposure to trading strategies we were able to ascertain that:

1. Stocks were analyzed singularly. That is, the behavior of the market was informally characterized for a single ticker and then various heuristic strategies were attempted to forecast the short term behavior of the selected ticker.

2. The identification of these "typical" graphically depicted behaviors is supposed to imply the behavior of the ticker in a short time span. Among them, the "Bollinger bands" and various configurations (sometimes referred to as "candles") based on the recent historical behavior of the selected ticker will lead to successful predictions. None of these heuristics was mathematically justified.

However, one of the things that convinced us of attempting to model ticket forecasting, anyway, was the realization that no formal analysis of the relation between one ticker and more than one has been published. The model we put forward explores the possibility that any ticker relies on the combined historical behavior of several ones. Implied in this assumption is the fact that the values of several tickers may better reflect the state of the stock market. Clearly this attempt to somehow correlate the behavior of one ticker to several others based on their previous history may or may not meet with success. But, in our opinion, it is worth discussing.

In this case of study you will be exposed to the full methodology. How to select a reasonable set of tickers, how to download the historical information from the Internet, how to set the forecasting databases, how to model the future behavior of a ticker given the past behavior of several others, how to quantify the results of the model. And, finally, the model's polynomials explicitly exhibit the mathematical relations between the tickers under consideration.

References

[1] Valdez-Valenzuela, E., Kuri-Morales, A., & Gomez-Adorno, H. (2024). Statistical Evaluation of Categorical Encoders for Pattern Preservation in Machine Learning Tasks. International Journal of Combinatorial Optimization Problems & Informatics, 15(2).

[2] Gilbert, J. K., Boulter, C., & Rutherford, M. (1998). Models in explanations, Part 1: Horses for courses?. *International journal of science education*, *20*(1), 83-97.

[3] Tolk, A. (2015). Learning something right from models that are wrong: Epistemology of simulation. In *Concepts and methodologies for modeling and simulation: A tribute to Tuncer Ören* (pp. 87-106). Cham: Springer International Publishing.

[4] Holst, A. (2021). Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. *Statista, June*, *1*(9).

[5] https://en.wikipedia.org/wiki/Boson, retrieved 10/14/24.

[6] Highfield, Roger (16 September 2008). "Large Hadron Collider: Thirteen ways to change the world". The Daily Telegraph. London. Archived from the original on 24 September 2009. Retrieved 10 October 2008.

[7] McPherson, S. S. (2009). *Tim Berners-Lee: Inventor of the World Wide Web*. Twenty-First Century Books.

[8] Van Der Meer, S. (1985). Stochastic cooling and the accumulation of antiprotons. *Reviews of Modern Physics*, *57*(3), 689.

[9] Jayawardhana, R. (2015). The Neutrino Hunters: The chase for the ghost particle and the secrets of the universe (softcover ed.).

[10] Raatikainen, Panu (2020). "Gödel's Incompleteness Theorems". Stanford Encyclopedia of Philosophy.

[11] Krogh, A. (2008). What are artificial neural networks?. *Nature biotechnology*, *26*(2), 195-197.

[12] Hemeida, A. M., Hassan, S. A., Mohamed, A. A. A., Alkhalaf, S., Mahmoud, M. M., Senjyu, T., & El-Din, A. B. (2020). Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research. *Ain Shams Engineering Journal*, *11*(3), 659-675.

[13] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. “Learning Internal Representations by Error Propagation”, Rumelhart, D., McClelland, J., and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.

[14] Haykin, Simon. Neural Networks and Learning Machines (3d Edition). Prentice Hall. ISBN-13: 978-0-13-147139-9, Chap. 4, Multilayer Perceptrons, 2009.

[15] M. J. D. Powell, “The theory of radial basis functions”, in Advances in Numerical Analysis II: Wavelets, Subdivision, and Radial Basis Functions, W. Light (eds), King Fahd University of Petroleum & Minerals, 1992.

[16] Haykin, Simon, op. cit, Chap. 5, Kernel Methods and Radial Basis Functions, 2009.

[17] Cortes, C. and Vapnik, V., "Support-Vector Networks", Machine Learning, 20, 1995.

[18] Haykin, Simon, op. cit, Chap. 6, Support Vector Machines, 2009.

[19] MacKay, D., Information Theory, Inference, and Learning Algorithms, Cambridge, ISBN 0-521-64298-1, (2004).

[20] Ratkowsky, D., Handbook of Nonlinear Regression Models, Marcel Dekker, Inc., New York, Library of Congress QA278.2 .R369, 1990.

[21] Beckermann, B., "The condition number of real Vandermonde, Krylov and positive definite Hankel matrices", Numerische Mathematik 85 (4): 553–577, doi:10.1007/PL00005392, 2000.

[22] Meyer, C., Matrix Analysis and Applied Linear Algebra, Society for Industrial and Applied Mathematics (SIAM), ISBN 978-0-89871-454-8, 2001.

[23]. Kuri-Morales, A. (2017). Closed determination of the number of neurons in the hidden layer of a multi-layered perceptron network. Soft Computing, 21, 597-609.

[24] Bishop, E., "A generalization of the Stone–Weierstrass theorem", Pacific Journal of Mathematics 11 (3): 777–783, 1961.

[25] McKinley, S., & Levine, M. (1998). Cubic spline interpolation. College of the Redwoods, 45(1), 1049-1060. J. H. AHLBERG, E. N. NILSON AND J. L. WALSH, The Theory of Splines and Their Applications, Academic Press, New York, 1967.

[26] Cybenko, G., "Approximation by superpositions of a sigmoidal function", Mathematics of Control, Signals and Systems, vol. 2, pp. 303-314, 1989.

[27] Koornwinder, T., Wong, R., Koekoek, R., Swarttouw, R., "Orthogonal Polynomials", in Olver, F., Lozier, D., Boisvert, R. et al., NIST Handbook of Mathematical Functions, Cambridge University Press, ISBN 978-0521192255, 2010.

[28] Cheney, E.W., Introduction to Approximation Theory, McGraw-Hill Book Company, pp. 34-45, 1966.

[29] Li, M., & Vitányi, P. (1988). Two decades of applied Kolmogorov complexity.

[30] https://archive.ics.uci.edu/ml/index.php

[31] Open Datasets and Machine Learning Projects | Kaggle,

https://www.kaggle.com/datasets.

[32] Kuri-Morales, A., Aldana-Bobadilla, E., "The Best Genetic Algorithm Part I", A Comparative Study of Structurally Different Genetic Algorithms, Springer, 1-15, Editor(s)): Castro, Gelbukh, González, ISBN: 9783642451119, ISSN: 1611-3349, 27/11/2013.

[33] Kuri-Morales, A., Aldana-Bobadilla, E., López-Peña, J., "The Best Genetic Algorithm Part II", A Comparative Study of Structurally Different Genetic Algorithms, Springer, 16-29, Editor(s)): Castro, Gelbukh, González, ISBN: 9783642451119, ISSN: 1611-3349, 27/11/2013.

[34] https://archive.ics.uci.edu/ml/datasets/wine, retrieved 11/11/2022

[35] Einstein, A. (1989). On the electrodynamics of moving bodies. The collected papers of Albert.

[36] https://www.youtube.com/watch?v=5eW6Eagr9XA&t=216s, “The 4 things it takes to be an expert”, retrieved 10/07/2024.

[37] https://en.wikipedia.org/wiki/Technical_analysis, “Technical Analysis”, retrieved 10/17/24.

[38] Saeed, R. (2020). Runge Phenomenon (Doctoral dissertation, School of Natural Sciences, National University of Sciences and Technology).

CHAPTER 2

UNIVARIATE COLLOCATION AND APPROXIMATION

2.1. Data Representation

Throughout this book we will assume that data is handled in a conventional binary digital computer [57]. We shall not consider quantum computers [53] or the associated quantum numbers. Furthermore, we shall not consider complex numbers [56]. We focus on real numbers in $\mathscr{R}$. A real number $\mathscr{R}$ represents an entity which has infinite precision. Because of the inherent finiteness of a digital computer's memory for computational purposes a real number must be replaced by a finite space to which we shall refer as $\mathscr{F}$. The number of variables will be denoted by N. Hence the models we will consider will lie in the hyperplane $\mathscr{F}^N$. The numbers in $\mathscr{F}$ are a subset of those in $\mathscr{R}$. This implies that the numbers in $\mathscr{F}$ are, themselves, a model. Ultimately, since we are assuming a digital computer, in actuality we consider variables in $\mathscr{B}$, the set of variables representable in binary.

The rough equivalent of a "real" number in a digital computer is a so-called floating point number (hence the $\mathscr{F}$ symbol). Depending on the hardware and the software the number of representable values may vary. The interested reader may look at the IEEE 754 standard [49]. As of 2025 the largest precision IEEE 754 format corresponds to 256-bit numbers. A very commonly used format is the extended precision IEEE 754 format illustrated in Fig. F2.1. It considers one bit for the sign of the significand, 15 bits for the exponent field and 64 bits for the significand. The exponent field is biased by 16383, meaning that 16383 has to be subtracted from the value in the exponent field to compute the actual power of 2. [54]

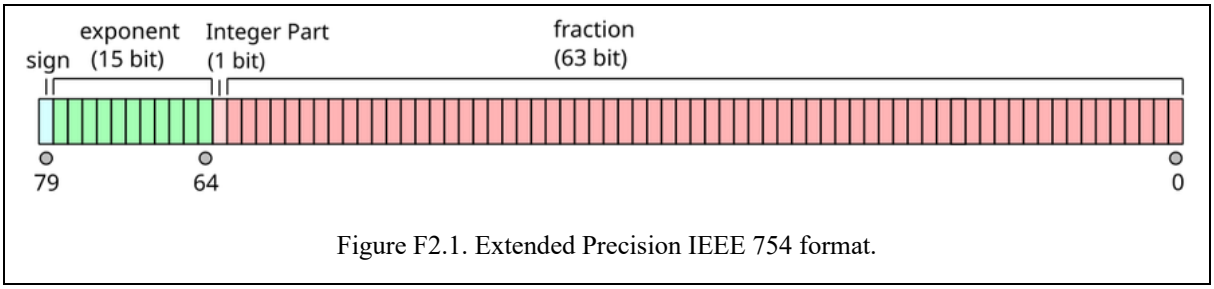

Figure F2.1. Extended Precision IEEE 754 format.

That means that with this format there are $2^{80} = (2^{10})^8 = (1{,}024)^8 \approx (10^3)^8 \approx 10^{24}$ possible representable numbers. This is a very large number but clearly finite. The 80-bit floating-point format has a range from approximately 3.65×10^{-4951} to $1.18 \times 10^{+4932}$. Although $\log_{10}(264) \approx 19.266$ this format is usually described as giving approximately eighteen significant digits of precision (the floor of $\log_{10}(263)$ which is the minimum guaranteed precision). The point is that, regardless of the format adopted in our hardware, we may not have a representation where two contiguous data points are more than $\approx 10^{-4951}$ units apart. In practice the precision of experimental data is very much larger than 10^{-4951}.

2.2. Collocation vs. Approximation

Once a representation format is assumed we now turn our attention to the issue of the meaning of a model's adequateness. The model's error for the *i-th* tuple, is $\varepsilon_i = f_i - y_i$, where f_i stands for the known values of tuple i and y_i stands for the value of f_i which the model yields. The global approximation error is denoted by ε. Given a set of experimental independent variables' values $\{x_1, x_2, \ldots, x_n\}$ and the corresponding experimental dependent variable's values $\{y_i\}$ we may either want to (a) Guarantee that $f(x_1, x_2, \ldots, x_n) = y_i \ \forall i$, i.e. our model yields an exact replica of the experimental dependent variable's values given the values of the independent variables, or (b) Guarantee that our model yields an approximate value of y such that the difference between y_i and f_i for all i satisfies a predefined criterion of adequateness such that $[f(x_1, x_2, \ldots, x_n) - y_i] \leq \varepsilon_{MAX}$. When the model is aimed at making $\varepsilon_i = (f_i - y_i) = 0 \quad \forall i$ we call such a model a Collocation Model. If, on the other hand, we accept the case where $\varepsilon \neq 0$, the model is called an Approximation Model. The goal of any Multivariate Approximation Algorithm is to find the values of the parameters in the function $Y = F(X_1, X_2, \ldots, X_N)$ such that the approximated values minimize an appropriately selected measure of error between the known values of the dependent variable Y and those

calculated from $F(X_1, X_2, ..., X_N)$ for ALL the objects in the experimental data. For the time being, we will focus on the univariate case $Y = F(X_1)$ and reserve the discussion of $Y = F(X_1, X_2, ..., X_N)$ for later on in the text.

2.2.1 Collocation

There are many ways in which to find a collocation model. Some of them will be examined in what follows.

2.2.1.1 Undetermined Coefficients

A very direct way to do find a collocation model is define a collocation polynomial of the form $y(x) = c_0 + c_1 x + c_2 x^2 + ... + c_n x^n$. In general,

$$y(x) = \sum_{i=0}^{n} c_i x^i \tag{1}$$

In order to obtain the c_{n+1} coefficients we need $n+1$ conditions. The values of the c_{n+1} may be gotten by solving the system of simultaneous equations in (2).

$$\begin{bmatrix} X_{00} & X_{01} & \dots & X_{0n} \\ X_{10} & X_{11} & \dots & X_{1n} \\ & & \cdots & \\ X_{n0} & X_{n1} & \dots & X_{nn} \end{bmatrix} \begin{bmatrix} c_0 \\ c_2 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix} \tag{2}$$

We can denote the system of (2) as the more compact vector product

$$\boldsymbol{XC}=\boldsymbol{y} \tag{3}$$

Where **X** is the matrix of the data, ***C*** is the vector of coefficients of the collocation polynomial and ***y*** is the vector of the dependent variable. Using Gaussian elimination (for which see section 2.2.1.1) the computational cost to find ***C*** is $O(n^3)$[6].

Another option, whose application's advantages will be discussed in chapter 4, is to find the coefficients ***C*** in (3) from $\boldsymbol{C} = \boldsymbol{By}$ where ***B*** is the inverse of ***X***, i.e. $\boldsymbol{B}=\boldsymbol{X}^{-1}$.

$$\boldsymbol{C} = \boldsymbol{By} \tag{4}$$

A variant of Gaussian elimination called Gauss–Jordan elimination can be used for finding the inverse of a matrix, if it exists. If **X** is an $n \times n$ square matrix, then one can use row reduction to compute its inverse matrix. First, the $n \times n$ identity matrix is augmented to the right of **X**, forming an $n \times 2n$ block matrix $[\boldsymbol{X} | \boldsymbol{I}]$, where ***I*** denotes the identity matrix. Through application of elementary row operations, find the reduced echelon form[7] of this $n \times 2n$ matrix. The matrix ***X*** is invertible if and only if the left block can be reduced to ***I***. In this case the right block of the final matrix is $\boldsymbol{X}^{-1}$. If the algorithm is unable to reduce the left block to ***I***, then ***X*** is not invertible. Once ***B*** is known the cost of calculating ***C*** using (4) is $O(n^2)$.

For example, assume we have the following data:

[6] The order of n or linear complexity, denoted by $O(n)$, is a notation we shall use to describe the rate of growth of a function as n increases, independently of the size of the inputs.

[7] A matrix is said to be in row echelon form when all its non-zero rows have a pivot, that is, a non-zero entry such that all the entries to its left and below it are equal to zero.

X	*Y*
1	2
3	5
6	3
8	4.5

Table T2.1. Data to Collocate in $\mathscr{R}^2$

The graph for the data in Table T2.1 is shown in Fig. F2.2.

To find the collocation polynomial we ought to select its degree. Since we have *n=4* data points the minimum degree is 3 (i.e. *n-1*). Tacit in this method is that, to conform to the polynomial in (1) the values of the independent variable X have to be mapped to the corresponding powers as illustrated in Table T2.2.

Therefore, the original problem in $\mathscr{R}^2$ is mapped into $\mathscr{R}^5$. Notice that the dependent variable is assumed to be in the last column of the tuple, a convention that we will follow throughout this book. In what follows we will denote the original dimension of the original data in T2.1 as *p* and the dimension of the mapped data in T2.2 as *q*.

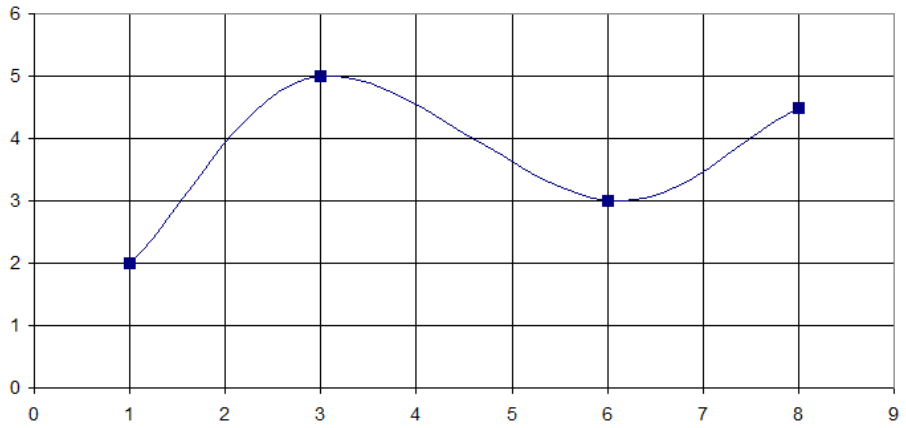

Figure F2.2. Graph for 4 Data Points

Hence:

x^0	x^1	x^2	x^3	*y*
1	1	1	1	2
1	3	9	27	5
1	6	36	216	3
1	8	64	512	4.5

Table T2.2. Data Mapped to $\mathscr{R}^5$.

We seek the coefficients which, when found, will yield a collocating polynomial such that, when input with the original values of x_i will output (with no error) the corresponding values of y_i.

In this case the coefficients in $y(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$ may be obtained by solving the system of simultaneous linear equations in (1) yielding $c_0 \approx$-2.642857, $c_1 \approx$+5.997619, $c_2 \approx$-1.457143. $c_3 \approx$+0.102381. To test the adequateness of the model we insert the values of $c_0, \ldots, c_3$ in the collocating function and get

FUNCTION	ORIGINAL
0) 2.000000	2.000000
1) 5.000000	5.000000
2) 3.000000	3.000000
3) 4.500000	4.500000

From which $f(x_i) = -2.642857 + 5.997619x - 1.457143\,x^2 + 0.102381\,x^3 \equiv y_i \;\forall i$.

2.2.1.1. Gaussian Elimination

There are many ways in which to solve the simultaneous linear equations in (1). LASSOL [73] implements the partial gaussian pivoting scheme. It is based on Gaussian elimination.

Gaussian Elimination [55] consists of a sequence of row-wise operations performed on the corresponding matrix of coefficients. This method can also be used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix.

2.2.1.1.1 Computational efficiency

The number of arithmetic operations required to perform row reduction is one way of measuring the algorithm's computational efficiency. For example, to solve a system of n equations for n unknowns by performing row operations on the matrix until it is in echelon form, and then solving for each unknown in reverse order, requires $(n^2+n)/2$ divisions, $(2n^3+3n^2-5n)/6$ multiplications, and $(2n^3+3n^2-5n)/6$ subtractions, for a total of approximately $2n^3/3$ operations. Thus it has an arithmetic complexity (time complexity, where each floating point arithmetic operation takes a unit of time, independently of the size of the inputs) of $O(n^3)$. This complexity is a good measure of the time needed for the whole computation when the time for each arithmetic operation is approximately constant. This is the case when the coefficients are represented by floating-point numbers.

2.2.1.2 Lagrange Interpolating Polynomial

Another way to find the coefficients in (1) is to appeal to a Lagrange Interpolating Polynomial. In numerical analysis, such polynomial is the unique polynomial of lowest degree that interpolates a given set of data.

Given a data set of coordinate pairs (x_j, y_j) with $0 \le j \le k$, the x_j are called nodes and the y_j are called values. The Lagrange polynomial $L(x)$ has degree $\le k$ and assumes each value at the corresponding node, $L(x_j) = y_j$.

Note, however, that for equispaced nodes, Lagrange interpolation is susceptible to Runge's phenomenon of large oscillation.

2.2.1.2.1 Definition

Given a set of $k+1$ nodes $\{x_0, x_1, \ldots, x_k\}$, which must all be distinct, $x_j \neq x_m$ for indices $j \neq m$, the Lagrange basis for polynomials of degree $\le k$ for those nodes is the set of polynomials $\{\ell_0(x), \ell_1(x), \ldots, \ell_k(x)\}$ each of degree k which take values $\ell_j(x_m)=0$ if $m \neq j$ and $\ell_j(x_j)=1$. Using the Kronecker delta this can be written $\ell_j(x_m)=\delta_{jm}$. Each basis polynomial can be explicitly described by the product:

$$\ell_j(x) = \frac{(x-x_0)}{(x_j-x_0)}\cdots\frac{(x-x_{j-1})}{(x_j-x_{j-1})}\frac{(x-x_{j+1})}{(x_j-x_{j+1})}\cdots\frac{(x-x_k)}{(x_j-x_k)}$$
$$= \prod_{\substack{0\le m<k \\ m\neq j}} \frac{x-x_m}{x_j-x_m} \quad .$$

The Lagrange interpolating polynomial for those nodes through the corresponding values $\{y_0, y_1, \ldots, y_k\}$ is the linear combination:

$$L(x) = \sum_{j=0}^{k} y_j \ell_j(x)$$

Each basis polynomial has degree k, so the sum $L(x)$ has degree $\leq k$, and it interpolates the data because $L(x_m) = \sum_{j=0}^{k} y_j \ell_j(x_m) = \sum_{j=0}^{k} y_j \delta_{mj} = y_m$.

Clearly the interpolating polynomial is unique, since we know that (1) has only one consistent solution. However, we may also (from Lagrange's polynomials) give the following proof. Assume the polynomial $M(x)$ of degree $\leq k$ interpolates the data. Then the difference $M(x)-L(x)$ is zero at $k+1$ distinct nodes $\{x_0,x_1, \ldots,x_k\}$. But the only polynomial of degree $\leq k$ with more than k roots is the constant zero function, so $M(x)-L(x)=0$, or $M(x)=L(x)$.

2.2.1.2.2. Runge's Phenomenon

Runge's phenomenon is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points. It shows that going to higher degrees does not always improve accuracy.

The Stone-Weierstrass Approximation Theorem (SWAT) states that for every continuous function $f(x)$ defined on an interval $[a,b]$, there exists a set of polynomial functions $P_n(x)$ for $n=0, 1, 2, \ldots$, each of degree at most n, that approximates $f(x)$ with uniform convergence over $[a,b]$ as n tends to infinity, that is,

$$\lim_{n\to\infty}\left(\sup_{a\leq x\leq b} | f(x) - P_n(x) | \right) = 0$$

Consider the case where one desires to interpolate through $n+1$ equispaced points of a function $f(x)$ using the n-degree polynomial $P_n(x)$ that passes through those points. One might expect from SWAT that using more points would lead to a more accurate reconstruction of $f(x)$. However, this particular set of polynomial functions $P_n(x)$ is not guaranteed to have the property of uniform convergence; the theorem only states that a set of polynomial functions exists, without providing a general method of finding one.

The $P_n(x)$ produced in this manner may in fact diverge away from $f(x)$ as n increases; this typically occurs in an oscillating pattern that magnifies near the ends of the interpolation points.

Consider the function $f(x)=1/(1+25x^2)$. If this function is interpolated at equidistant points x_i between -1 and 1 such that $x_i = (2i/n) - 1 \;\; i \in \{0,1,\ldots,n\}$ with a polynomial $P_n(x)$ of degree $\leq n$ the resulting interpolation oscillates toward the end of the interval, i.e. close to -1 and 1. It can even be proven that the interpolation error increases (without bound) when the degree of the polynomial is increased.

In Figure F2.3 we illustrate Rung's phenomenon stemming from the function $y=1/(1+25\ x^2)$.

Runge's phenomenon is the consequence of two properties of this problem.

a) The magnitude of the n-th order derivatives of this particular function grows quickly when n increases.

b) The equidistance between points leads to a Lebesgue constant that increases quickly when n increases.

The phenomenon is graphically obvious because both properties combine to increase the magnitude of the oscillations. Although often used to explain Runge´s Phenomenon, the fact that the upper bound of the error goes to infinity does not necessarily imply, of course, that the error itself also diverges with n.

The problem can be avoided by using spline curves which are piecewise polynomials. When trying to decrease the interpolation error one can increase the number of polynomial pieces which are used to construct the spline instead of increasing the degree of the polynomials used.

2.2.1.3 Splines

"The term 'spline' is used to refer to a wide class of functions that are used in applications requiring data interpolation and/or smoothing. The data may be either one-dimensional or multi-dimensional. Spline functions for interpolation are normally determined as the minimizers of suitable measures of roughness (for example integral squared curvature) subject to the interpolation constraints. Smoothing splines may be viewed as generalizations of interpolation splines where the functions are determined to minimize a weighted combination of the average squared approximation error over observed data and the roughness measure. For a number of meaningful definitions of the roughness measure, the spline functions are found to be finite dimensional in nature, which is the primary reason for their utility in

computations and representation. For the rest of this section, we focus entirely on one-dimensional, polynomial splines and use the term 'spline' in this restricted sense." [59] [60]

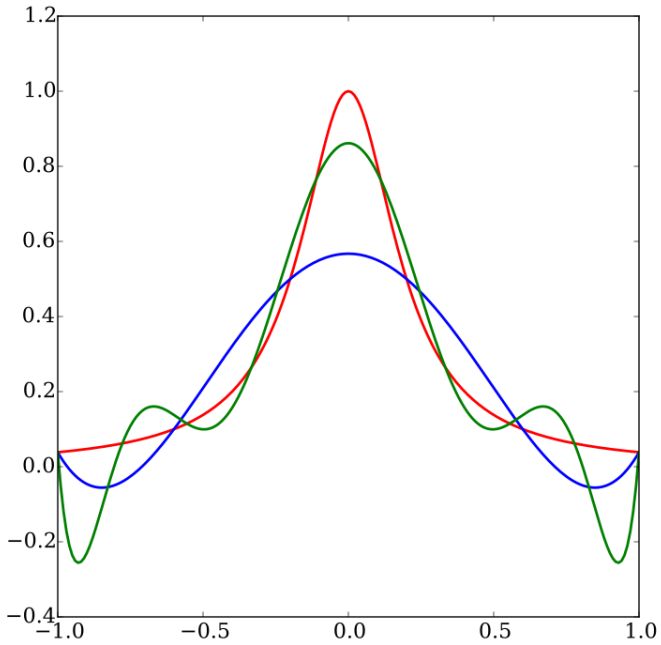

Figure F2.3. Illustration of the Runge phenomenon in which the oscillations near the interval's boundaries increase with higher degree polynomial interpolations

- The function $1/(1+25\ x^2)$
- A fifth degree polynomial interpolation (exact replication of the red curve at 6 points)
- A ninth degree polynomial interpolation (exact replication of the red curve at 10 points)

2.2.1.3.1. Definition of a Natural Spline

We now define a set of cubic polynomials *S(x)* which we call a "spline" as

$$S(x) = \sum_{i=0}^{n-1} (a_{i,i+1} + b_{i,i+1}x + c_{i,i+1}x^2 + d_{i,i+1}x^3)\delta(x)$$

$$\delta(x) = \begin{cases} 1 & x_i \le x \le x_{i+1} \\ 0 & otherwise \end{cases}$$

We denote the i-th cubic polynomial with $S_i(x)$. These polynomials join the "knots" of the spline in such a way that:

a) The spline collocates the original data. Hence,

$$S(x_i) = f(x_i), \quad i = 0,1,...,n \tag{2.1}$$

b) The spline is continuous on its first derivative. That is,

$$S_i'(x_i) = S_{i+1}'(x_i); \quad i = 1,2,...,n-2 \tag{2.2}$$

c) The spline is also continuous on its second derivative:

$$S_i''(x_i) = S_{i+1}''(x_i); \quad i = 1,2,...,n-2 \tag{2.3}$$

d) The second derivative of the spline at its end points is zero:

$$S''(x_0) = S''(x_n) = 0 \tag{2.4}$$

Condition (d) gives rise to the so-called “natural” spline.

□

Consider again the data of table T2.1. The corresponding spline is illustrated in Figure F2.4.

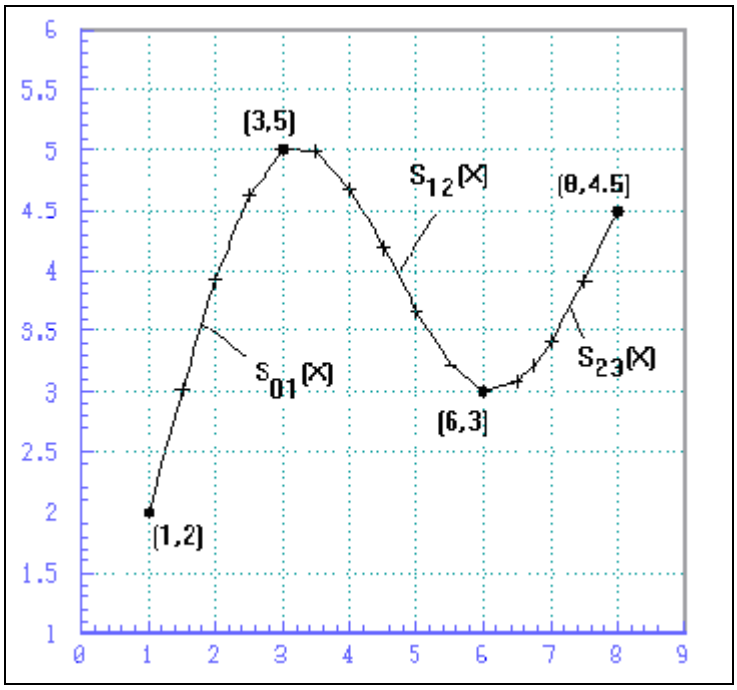

Figure F2.4. An Example of a Spline

The elements of the spline are of degree 3 because this is the smallest degree that allows us to comply with all of the conditions above, as will become clear from what follows. In the figure, the polynomials are given by

$$S_{01}(x) = a_{01} + b_{01}x + c_{01}x^2 + d_{01}x^3$$
$$S_{12}(x) = a_{12} + b_{12}x + c_{12}x^2 + d_{12}x^3$$
$$S_{23}(x) = a_{23} + b_{23}x + c_{23}x^2 + d_{23}x^3$$

Finding the spline means “to determine the set of coefficients” above.

From the collocating condition we may write:

$$a_{01} + b_{01}x_0 + c_{01}x_0^{\ 2} + d_{01}x_0^{\ 3} = y_0$$
$$a_{01} + b_{01}x_1 + c_{01}x_1^{\ 2} + d_{01}x_1^{\ 3} = y_1$$

which is a system with 4 unknowns and only two conditions. We may establish two further conditions from the fact that the spline is required to be continuous for its first and second derivatives. The first and second derivatives are given by:

$$b_{i,i+1} + 2c_{i,i+1}x + 3d_{i,i+1}x^2 = y'(x)$$
$$2c_{i,i+1} + 6d_{i,i+1}x = y''(x)$$

Then

$$b_{01} + 2c_{01}x_1 + 3d_{01}x_1^{\ 2} = b_{12} + 2c_{12}x_1 + 3d_{12}x_1^{\ 2}$$

and

$$2c_{01} + 6d_{01}x_1 = 2c_{12} + 6d_{12}x_1$$

For the second cubic we also know that

$$a_{12} + b_{12}x_1 + c_{12}x_1^{\ 2} + d_{12}x_1^{\ 3} = y_1$$
$$a_{12} + b_{12}x_2 + c_{12}x_2^{\ 2} + d_{12}x_2^{\ 3} = y_2$$

Now we have 6 conditions but 8 unknowns. It is easy to see that, as we continue in this fashion, there will always be *n* conditions and *n+2* unknowns. Therefore, to be able to completely define the spline, we must stipulate two more conditions. These may be defined in several ways. Here we only consider (for reasons that will be explained in what follows) that

$$S''(x_0) = 0 \text{ and } S''(x_n) = 0$$

and we may solve the resulting system of linear equations by any of the known methods. This system consists of *4(n+1)* linear equations whose solution yields *4(n+1)* defining coefficients.

2.2.1.3.2 Advantages of Natural Splines

It would seem that the spline is an uneconomical way to handle a collocation problem, since a single polynomial requires only *n+1* equations and the corresponding *n+1* coefficients. In what follows we develop a method which deems it unnecessary to solve the set of simultaneous linear equations and, furthermore, allows us to find the interpolated points without the need for an explicit calculation of the coefficients of the equations. Rather, we find the values of the second derivatives at the data points. Although not needed, should we decide to do so, we may find the *4(n+1)* coefficients from the *n+1* second derivatives.

For convenience, let

$$h_i = x_{i+1} - x_i$$
$$s_i = S''(x_i)$$

From the stipulation that *S(x)* be twice continuously differentiable we may write:

$$S_i''(x) = s_i \frac{x_{i+1} - x}{h_i} + s_{i+1} \frac{x - x_i}{h_i} \tag{2.5}$$

Integrating (5) twice, we get

$$S_i(x) = \frac{s_i}{6h_i}(x_{i+1} - x)^3 + \frac{s_{i+1}}{6h_i}(x - x_i)^3 + c_1(x - x_i) + c_2(x_{i+1} - x) \tag{2.6}$$

From the collocation condition we have

$$S_i(x_i) = f_i$$
$$S_i(x_{i+1}) = f_{i+1}$$

from which we find c_1 and c_2 and write:

$$S_i(x) = \frac{s_i}{6h_i}(x_{i+1} - x)^3 + \frac{s_{i+1}}{6h_i}(x - x_i)^3 + \left(\frac{f_{i+1}}{h_i} - \frac{s_{i+1}h_i}{6}\right)(x - x_i) + \left(\frac{f_i}{h_i} - \frac{s_i h_i}{6}\right)(x_{i+1} - x) \tag{2.7}$$

Differentiating once with respect to x:

$$S_i'(x) = -\frac{s_i}{2h_i}(x_{i+1} - x)^2 + \frac{s_{i+1}}{2h_i}(x - x_i)^2 + \frac{f_{i+1} - f_i}{h_i} - \frac{h_i}{6}(s_{i+1} - s_i) \tag{2.8}$$

Since we require that the spline be continuous for its first derivative, we know that:

$$S_{i-1}'(x) = S_i'(x)$$

Therefore,

$$s_{i+1} + 2s_i\left(\frac{h_i + h_{i-1}}{h_i}\right) + \frac{h_{i-1}}{h_i}s_{i-1} = \frac{6}{h_i}\left(\frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}\right) \tag{2.9}$$

$$i=1,2,\ldots,n-1$$

Expression (2.9) defines a system of $n+1$ unknowns ($s_0, s_1, \ldots, s_n$) with $n-1$ conditions. We need two additional conditions which we chose (for reasons that will be discussed later) to be:

$$S''(x_0) = S''(x_n) = 0 \quad \longrightarrow \quad s_0 = s_n = 0 \tag{2.10}$$

2.2.1.3.3 Calculation of the Natural Spline

We re-write expression (2.9) as:

$$\frac{h_{i-1}}{h_i}s_{i-1} + 2s_i\left(1 + \frac{h_{i-1}}{h_i}\right)s_i + s_{i+1} = d_i$$

$$i=1,2,\ldots,n-1 \tag{2.11}$$

where

$$d_i = \frac{6}{h_i}\left(\frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}\right)$$

We also define

$$s_{i-1} = \sigma_i s_i + \tau_i; \quad i = 1,2,\ldots,n \tag{2.12}$$

Since $s_0 = 0$ then

$$\sigma_1 = \tau_1 = 0 \tag{2.13}$$

Putting (2.11) in (2.12), we have:

$$s_i = \frac{-1}{\frac{h_{i-1}}{h_i}\sigma_i + 2\left(1 + \frac{h_{i-1}}{h_i}\right)}s_{i+1} + \frac{d_i - \frac{h_{i-1}}{h_i}\tau_i}{\frac{h_{i-1}}{h_i}\sigma_i + 2\left(1 + \frac{h_{i-1}}{h_i}\right)} \tag{2.14}$$

Equation (2.14) has the same form as equation (2.12). We may, therefore, write:

$$\sigma_{i+1} = \frac{-1}{\frac{h_{i-1}}{h_i}\sigma_i + 2\left(1+\frac{h_{i-1}}{h_i}\right)}; \quad \tau_{i+1} = \frac{d_i - \frac{h_{i-1}}{h_i}\tau_i}{\frac{h_{i-1}}{h_i}\sigma_i + 2\left(1+\frac{h_{i-1}}{h_i}\right)} \tag{2.15}$$

From which we may solve σ_i, τ_i for *i=1,2,...,n* since we already know [from (2.13)] that $\sigma_1 = \tau_1 = 0$. From these values we may find the s_i in (2.14) starting with $s_n = 0$ and then working backwards for *i=n-1*, *n-2*, ..., *0*. Once the s_i are known, we may find the interpolated sought for values directly from (2.7).

$$S_i(x) = \frac{s_i}{6h_i}(x_{i+1}-x)^3 + \frac{s_{i+1}}{6h_i}(x-x_i)^3 + \left(\frac{f_{i+1}}{h_i} - \frac{s_{i+1}h_i}{6}\right)(x-x_i) + \left(\frac{f_i}{h_i} - \frac{s_i h_i}{6}\right)(x_{i+1}-x) \tag{2.16}$$

and we need not bother to find the coefficients for the cubics $S_i(x)$, although we may do so if desired.

2.2.1.3.4. Minimum Curvature Property of Natural Splines

The conditions in (10) seem to be arbitrary. However, these conditions guarantee that the interpolant is the twice continuously differentiable one which approximates the data points with the least possible "curvature".

Proof.

Let S(x) be the collocation spline; let g(x) be any other collocation function. Also let *f(x)* be the function being approximated.

We may write:

$$\begin{aligned}\int_a^b [g''(t)-S''(t)]^2 dt = &\int_a^b [g''(t)]^2 dt - \\ &2\int_a^b [g''(t)-S''(t)]S''(t)dt - \\ &\int_a^b [S''(t)]^2 dt\end{aligned} \tag{2.17}$$

since $a = x_0 < x_1 < ... < x_n = b$, we may express the integral from the second term to the right of the "=" sign (which we denote by T) as:

$$T = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} S''(t)[g''(t)-S''(t)]dt \tag{2.18}$$

Integration by parts ($\int u\,dv = uv - \int v\,du$) yields:

$$T = \sum_{i=0}^{n-1} \{[g'(t)-S'(t)]S''(t)\Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} [g'(t)-S'(t)]S'''(t)dt\}$$

Since S'''(t) on $[x_i, x_{i+1}]$ is constant (call it α_i) we may write:

$$\int_{x_i}^{x_{i+1}} [g'(t)-S'(t)]S'''(t)dt = \alpha_i [g(t)-S(t)]\Big|_{x_i}^{x_{i+1}} \tag{2.19}$$

which vanishes, since

$$g(x_i) - S(x_i) = f(x_i) - f(x_i) = 0$$

Hence,

$$T = \sum_{i=0}^{n-1} \{S''(x_{i+1})[g'(x_{i+1}) - S'(x_{i+1})] - S''(x_i)[g'(x_i) - S'(x_i)]\} \tag{2.20}$$

This sum reduces to the extreme values:

$$T = S''(x_n)[g'(x_n - S'(x_n)] - S''(x_0)[g'(x_0) - S'(x_0)] \tag{2.21}$$

If we choose (as suggested) $S''(x_n) = S''(x_0) = 0$, then these last two terms of (2.21) vanish as well, and we have that

$$T = 0 \tag{2.22}$$

We may then write (2.17) as:

$$\int_a^b [g''(t) - S''(t)]^2 \, dt = \int_a^b [g''(t)]^2 \, dt - \int_a^b [S''(t)]^2 \, dt \tag{2.23}$$

and

$$\int_a^b [g''(t)]^2 \, dt = \int_a^b [S''(t)]^2 \, dt + \int_a^b [g''(t) - S''(t)]^2 \, dt \tag{2.24}$$

From which finally we conclude that

$$\int_a^b [g''(t)]^2 \, dt \geq \int_a^b [S''(t)]^2 \, dt \tag{2.25}$$

Thus, **from all twice continuously differentiable collocating functions, the spline with $s_0 = s_n = 0$ has the least value for the integral in (2.25)**. Intuitively, what we have found means that while a straight line would give its minimum value to the integral, *S(x)* is as close to a straight line as allowed by collocation. Here we emphasize that spline interpolation be simply calculated from a procedure and a function shown in Pseudo-codes P2.1. and P2.2. We need to run SPCOEF only once; SPLNE is run once for every point to interpolate. As shown, the code is compact and fast. It yields the *n-2* second derivatives (s_0 and s_n are always 0) of the spline and the required interpolated values.

2.2.1.3.5 Implementation

```
(σ and τ are auxiliary storage vectors)
σ 2 ← 0
τ 2 ← 0
for i ← 2 to n-1
    hi-1 ← xi-xi-1
    hi ← xi+1-xi
    temp ← (hi-1/hi) ( σ i+2)+2
    σ i+1 ← -1/ temp
    d=6 [(yi+1-yi)/hi - (yi-yi-1)/hi-1] / hi
    τ i+1 ← (d-hi-1 τ i/hi) / temp
endfor
s1 ← 0
sn ← 0
for i ← 1 to n-2
    iB ← n-i
    siB ← σ iB+1siB+1+ τ iB+1
endfor
```

Pseudo Code P2.1: Procedure SPCOEF