ScienceDirect

Reduction Algorithm

Related terms:

Dimensionality Reduction, Helmholtz Equation, Poisson Equation, Eigenvalues, Eigenvector, Modulo

View all Topics

Modular Reduction

Tom St Denis, Greg Rose, in BigNum Math, 2006

6.3.2 Baseline Montgomery Reduction

The baseline Montgomery reduction algorithm will produce the residue for any size input. It is designed to be a catch-all algorithm for Montgomery reductions.

Algorithm mp_montgomery_reduce. This algorithm reduces the input x modulo n in place using the Montgomery reduction algorithm. The algorithm is loosely based on algorithm 14.32 of [2, pp.601], except it merges the multiplication of μn□t with the addition in the inner loop. The restrictions on this algorithm are fairly easy to adapt to. First, $0 \le x < n^2$ bounds the input to numbers in the same range as for the Barrett algorithm. Additionally, if $n>1$ and n is odd there will exist a modular inverse □. □ must be calculated in advance of this algorithm. Finally, the variable k is fixed and a pseudonym for *n.used*(Figure 6.9).

Figure 6.9. Algorithm mp_montgomery_reduce

Step 2 decides whether a faster Montgomery algorithm can be used. It is based on the Comba technique, meaning that there are limits on the size of the input. This algorithm is discussed in 7.9.

Step 5 is the main reduction loop of the algorithm. The value of μ is calculated once per iteration in the outer loop. The inner loop calculates $x + \mu n$􏰁$ix$by multiplying μn and adding the result to x shifted by ix digits. Both the addition and multiplication are performed in the same loop to save time and memory. Step 5.4 will handle any additional carries that escape the inner loop.

On quick inspection, this algorithm requires n single precision multiplications for the outer loop and n^2 single precision multiplications in the inner loop for a total $n^2 + n$ single precision multiplications, which compares favorably to Barrett at $n^2 + 2n-1$ single precision multiplications.

This is the baseline implementation of the Montgomery reduction algorithm. Lines 31 to 36 determine if the Comba–based routine can be used instead. Line 47 computes the value of μ for that particular iteration of the outer loop.

The multiplication *μn*□*ix*is performed in one step in the inner loop. The alias *tmpx* refers to the *ix*□th digit of *x*, and the alias *tmpn* refers to the modulus *n*.

> Read full chapter

Interactive Design Optimization

Jasbir S. Arora, in Introduction to Optimum Design (Second Edition), 2004

13.2.1 Cost Reduction Algorithm

A subproblem for the cost reduction algorithm can be defined with or without the approximate Hessian **H**. Without Hessian updating, the problem is defined in Eqs. (10.25) and (10.26) and, with Hessian updating, it is defined in Eqs. (11.48) to

(11.50). Although Hessian updating can be used, we shall define the cost reduction subproblem without it to keep the discussion and the presentation simple. Since the cost reduction problem is solved from a feasible or almost feasible point, the right side vector **e** in Eq. (10.26) is zero. Thus, the cost reduction QP subproblem is defined as

(13.1)

subject to

(13.2)

(13.3)

The columns of matrices **N** and **A** contain gradients of equality and inequality constraints, respectively, and **c** is the gradient of the cost function. Equation (13.2) gives the dot product of d with all the columns of **N** as zero. Therefore, **d** is orthogonal to the gradients of all the equality constraints. Since gradients in the matrix **N** are normal to the corresponding constraint surfaces, the search direction **d** lies in a plane tangent to the equality constraints. The right side vector **b** for the inequality constraints in Eq. (13.3) contains zero elements corresponding to the active constraints and positive elements corresponding to the inactive constraints. If an active constraint remains satisfied at the equality (i.e., $\mathbf{a}_{(i)} \cdot \mathbf{d} = 0$), the direction **d** is in a plane tangent to that constraint. Otherwise, it must point into the feasible region for the constraint.

The QP subproblem defined in Eqs. (13.1) to (13.3) can incorporate the potential constraint strategy as explained in Section 11.1. The subproblem can be solved for the cost reduction direction by any of the available subroutines cited in Section 11.2. In the example problems, however, we shall solve the QP subproblem using KKT conditions. We shall call this procedure of reducing cost from a feasible point the *cost reduction (CR) algorithm.*

After the direction has been determined, the step size can be calculated by a line search on the proper descent function. Or, we can require a certain reduction in the cost function and determine the step size that way. For example, we can require a fractional reduction □ in the cost function (for a 5 percent reduction, □ = 0.05), and calculate a step size based on it. Let □ be the step size along **d**. Then the first-order change in the cost using a linear Taylor's expansion is given as □$|\mathbf{c} \cdot \mathbf{d}|$. Equating this to the required reduction in cost $|$□$f|$, the step size is calculated as

(13.4)

Note that **c**˙**d** should not be zero in Eq. (13.4) to give a reasonable step size. The cost reduction step is illustrated in Example 13.1.

EXAMPLE 13.1

Cost Reduction Step

Consider the design optimization problem

subject to

From the feasible point (4, 4), calculate the cost reduction direction and the new design point requiring a cost reduction of 10 percent.

Solution. The constraints can be written in the standard form as

The optimum solution for the problem is calculated using the KKT conditions as

At the given point (4, 4),

Therefore, constraint g_2 is active, and all the others are inactive. The cost function is much larger than the optimum value. The constraints for the problem are plotted in Fig. 13-2. The feasible region is identified as 0ABC. Several cost function contours are shown there. The optimum solution is at the point B (6, 3). The given point (4, 4) is identified as D on the line B–C in Fig. 13-2.

The gradients of cost and constraint functions at the point D (4, 4) are calculated as

These gradients are shown at point D in Fig. 13-2. Each constraint gradient points to the direction in which the constraint function value increases. Using these quantities, the QP subproblem of Eqs. (13.1) to (13.3) is defined as

subiect to

The solution for the QP subproblem using KKT conditions or the Simplex method of Section 10.4 is

d = (-0.5, – 3.5); **u** = (43.5, 0, 0, 0)

At the solution, only the first constrain is active having a positive Lagrange multiplier. The direction **d** is shown in Fig. 13-2. Since the second constraint is inactive, $\mathbf{a}_{(2)}$˙**d** must be negative according to Eq. (13.3) and it is (-0.625). Therefore, direction **d**points toward the feasible region with respect to the second constraint, which can be observed in Fig. 13-2.

The step size is calculated from Eq. (13.4) based on a 10 percent reduction (gM = 0.1) of the cost function as

Thus, the new design point is given as

which is quite close to point D along direction **d**. The cost function at this point is calculated as

$$f(x^{(1)})=-26.304$$

which is approximately 10 percent less than the one at the current point (4, 4). It may be checked that all constraints are inactive at the new point.

Direction **d** points into the feasible region at point **D** as can be seen in Fig. 13-2. Any small move along **d** results in a feasible design. If the step size is taken as 1 (which would be obtained if the inaccurate line search of Section 11.3 was performed), then the new point is given as (3.5, 0.5), which is marked as E in Fig. 13-2. At point E, constraint g_1 is active and the cost function has a value of –29.875, which is smaller than the previous value of –26.304. If we perform an exact line search, then □ is computed as 0.5586 and the new point is given as (3.7207, 2.0449)—identified as point E' in Fig. 13-2. The cost function at this point is –39.641, which is still better than the one with step size as unity.

> Read full chapter

Computer Solution of Large Linear Systems

In Studies in Mathematics and Its Applications, 1999

3.6.2 The Multilevel Spectral Algorithm

Even if the eigenvalue and eigenvector do not have to be computed very accurately, the spectral algorithm as formulated above is too costly compared to other profile reduction algorithms for large examples. To overcome this problem, Barnard, Pothen and Simon proposed using the algorithm on a contracted graph with much fewer vertices than G:
□□□□□□□

1) construct a series of coarser and coarser graphs that retains the structure of the original graph,
2) compute the second eigenvector of the coarsest graph,
3) interpolate this vector to the next finer graph,
4) refine the interpolated vector (by Rayleigh Quotient Iteration, see Parlett [-1062]) and go to 3) until we are back to the original graph.

There are many ways to define graph contraction. The one proposed by Barnard, Pothen and Simon was to find a maximal independent set of vertices which are to be the vertices of the contraction. The edges are found by growing domains from the selected vertices adding an edge when two domains intersect. Good results were reported in [116]. A similar algorithm has been introduced independently by Paulino, Menezes, Gattass and Mukherjee [1068].

> Read full chapter

Feature Selection and Dimensionality Reduction

In Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications, 2012

Linear Dimensionality Reduction Approaches

The rest of this chapter is devoted to linear dimensionality reduction techniques, since they are by far the most widely used in text mining today. We discuss the most popular approaches, including the following:

- Latent semantic indexing (LSI)
- Principal components analysis (PCA)
- Factor analysis (FA)
- Projection pursuit

Latent Semantic Indexing

LSI is a linear dimensionality reduction technique that operates on the term-document matrix. LSI uses a common matrix operation called the singular value decomposition to identify independent components of the data. LSI is one of the most popular dimensionality reduction algorithms for text because it often provides dimensions with semantic meaning; features in the same dimension are often topically related. Chapter 15 provides an in-depth discussion of the singular value decomposition (SVD) for LSI.

Principal Components Analysis

Like LSI, PCA relies on the SVD, but PCA operates on the covariance or correlation coefficient matrix between features, not on the term-document matrix. PCA was created by Karl Pearson (1901) for the purposes of transforming a data set with possibly many correlated variables into a simpler data set with fewer but uncorrelated variables. For many data sets, the largest principal components capture most of the relevant information in the data, allowing the other components to be discarded (Fodor, 2002). Because the components are based on feature correlation and not document co-occurrence (like LSI), the components extracted by PCA are not easily interpretable. For some applications, this makes PCA less desirable, despite its effectiveness at reducing the dimensionality of the data without harming performance.

Factor Analysis

FA comes from the field of psychology where it is used to identify variables that may have been caused by a shared, unobserved "factor" (Fodor, 2002). FA operates on the covariance matrix between features and uses matrix arithmetic to identify overlapping features. It is a second-order, linear method.

Projection Pursuit

Unlike the preceding methods, projection pursuit is a linear method that can incorporate more than second-order information, making it well suited for data that are not normally distributed (Fodor, 2002). Projection pursuit uses some measure of quality to determine which projections of the data to pursue. It can be thought of as a generalization of PCA, since running projection pursuit on the second-order variance matrix produces the same results as PCA.

> Read full chapter

Getting Started

Tom St Denis, Greg Rose, in BigNum Math, 2006

2.1 Library Basics

The trick to writing any useful library of source code is to build a solid foundation and work outward from it. First, a problem along with allowable solution parameters should be identified and analyzed. In this particular case, the inability to accommodate multiple precision integers is the problem. Furthermore, the solution must be written as portable source code that is reasonably efficient across several different computer platforms.

After a foundation is formed, the remainder of the library can be designed and implemented in a hierarchical fashion. That is, to implement the lowest level dependencies first and work toward the most abstract functions last. For example, before implementing a modular exponentiation algorithm, one would implement a modular reduction algorithm. By building outward from a base foundation instead of using a parallel design methodology, you end up with a project that is highly modular. Being highly modular is a desirable property of any project as it often means the resulting product has a small footprint and updates are easy to perform.

Usually, when I start a project I will begin with the header files. I define the data types I think I will need and prototype the initial functions that are not dependent on other functions (within the library). After I implement these base functions, I prototype more dependent functions and implement them. The process repeats until I implement all the functions I require. For example, in the case of LibTomMath, I implemented functions such as mp_init() well before I implemented mp_mul(), and even further before I implemented mp_exptmod(). As an example as to why this design works, note that the Karatsuba and Toom-Cook multipliers were written *after* the dependent function mp_exptmod() was written. Adding the new multiplication algorithms did not require changes to the mp_exptmod() function itself and lowered the total cost of ownership and development (*so to speak*) for new algorithms. This methodology allows new algorithms to be tested in a complete framework with relative ease(Figure 2.1).

Figure 2.1:. Design Flow of the First Few Original LibTomMath Functions.

Only after the majority of the functions were in place did I pursue a less hierarchical approach to auditing and optimizing the source code. For example, one day I may audit the multipliers and the next day the polynomial basis functions.

It only makes sense to begin the text with the preliminary data types and support algorithms required. This chapter discusses the core algorithms of the library that are the dependents for every other algorithm.

> Read full chapter

Optical Radiometry for Ocean Climate Measurements

Giuseppe Zibordi, Kenneth J. Voss, in Experimental Methods in the Physical Sciences, 2014

1 Introduction

Decadal time-series of Earth Observation (EO) data from multiple missions are a unique source of climate variables at different spatial scales. Nevertheless, the successful application of EO data to the detection and quantification of small trends embedded in large natural variations requires traceability to the International System of Units and, high accuracy and consistency over time of these data products. In the case of satellite ocean color, this can only be met through (1) extraordinary prelaunch sensor characterization and calibration; (2) on-orbit tracking of sensor radiometric stability (i.e., accuracy variation with time) and adjustment of responsivity changes; (3) indirect calibration of combined sensor responsivity and data reduction algorithms (i.e., system vicarious calibration); and (4) assessment (i.e., validation) of data products by quantifying statistical indices (e.g., bias and dispersion). The fulfillment of these tasks requires the definition and implementation of calibration and validation programs lasting beyond the lifetime of individual EO missions.

An essential component of ocean-color calibration and validation programs is the collection of high-quality in situ data, which are central for system vicarious calibration, validation of data products, and additionally development and assessment of bio-optical models for the generation of derived high-level products. Because of this, the collection and handling of field data have an important position in postlaunch EO strategies. For instance, in situ data supporting postlaunch system vicarious calibration must be of extremely high quality to allow accurate simulation of the radiometric signal at the EO sensor for site(s) representative of the most common marine and atmospheric conditions. These in situ data are generally produced through unique long-term deployment buoy-based systems with the aid of state-of-the-art methods for the characterization and calibration of field instruments, for data collection, and finally for data reduction and quality control. In addition to system vicarious calibration, validation and bio-optical modeling activities require in situ data representative of the multitude of observation conditions covering the variety of world marine water types. Because of this, validation data sets are commonly constructed by combining measurements performed with a number of instruments operated by independent teams on a variety of deployment platforms (i.e., ships, buoys, and offshore fixed structures). Thus, despite the effort to enforce the use of community protocols for the characterization and calibration of in situ instruments, and additionally for the collection, reduction, and quality control of data, it is unlikely that validation data sets can exhibit the same traceability, accuracy, and consistency as in situ measurements specifically produced to support system vicarious calibration.

This chapter discusses requirements for in situ optical radiometry data supporting satellite ocean-color missions targeted to climate-change applications. Emphasis is placed on basic strategies relevant to measurement programs for system vicarious calibration and validation of data products pursuing the principle that *adequately sampled, carefully calibrated, quality-controlled, and archived data for key elements of the climate system will be useful indefinitely* [1].

> Read full chapter

Tree-Adapted Wavelet Shrinkage

James S. Walker, in Advances in Imaging and Electron Physics, 2002

C The ASWDR ALGORITHM

The Taws algorithm combines the three Taws selection principles with the Aswdr image-compression algorithm. Hence, before Taws is described, Aswdr must first be summarized.

The Aswdr image-compression algorithm consists of five parts, as shown in Figure 10. In the initialization part of Aswdr, a wavelet transform of theimage is computed. An initial threshold value T_0 is chosen so that all transform values have magnitudes that are less than $2T_0$ and at least one has magnitude greater than or equal to T_0. The purpose of the loop indicated in Figure 10 is to encode significant transform values by the method of *bit-plane encoding*. A binary expansion, relative to the quantity $2T_0$, is computed for each transform value. The loop constitutes the procedure by which these binary expansions are calculated. As the threshold is successively reduced by half, the parts labeled *significance pass* and *refinement pass* compute the next bit in the binary expansions of the transform values.* It will be seen that replacing the threshold T by its half-value $T/2$, along with the looping through the significance pass, results in a logically consistent method for testing the significance of parents and children.

Figure 10. Block diagram for Aswdr and Taws algorithms.

Each part of the Aswdr algorithm is next described in more detail. The initialization part, as described previously, involves wavelet transforming the image and choosing an initial threshold $T = T_0$. One other task in initialization is assigning a *scan order*. For an image with P pixels, a scan order is a one-to-one and onto mapping, , for $k = 1, \dots P$, between the transform values and a linear ordering (x_k). This initial scan order is a zigzag from higher to lower levels (Shapiro, 1993), with row-based scanning through the horizontal coefficients, column-based scanning through the vertical coefficients, and zigzag scanning through the diagonal coefficients.

The next part of the algorithm is the significance pass. In this part, *new* significant transform values x_m satisfying $T \leq |x_m| < 2T$ are identified. Their index values m are encoded by using the *difference reduction method* of Tian and Wells (1996). The difference reduction method essentially consists of a binary encoding of the number of steps to go from the index of the last significant value to the index of the present significant value. More details can be found in Tian and Wells (1998) or Walker and Nguyen (2000b). The *quantized* value $q_m = T\,\text{sgn}(x_m)$ is assigned to the index m at this point.

Following the significance pass, there is a refinement pass. The refinement pass is a process of refining the precision of *old* quantized transform values q_n, which satisfy $|q_n| \geq 2T$. Each refined value is a better approximation of an exact transform value. The precision of quantized values is increased to make them at least as accurate as the present threshold. For example, if an old significant transform value's magnitude $|x_n|$ lies in the interval [32, 48), say, and the present threshold is 8, then whether its magnitude lies in [32, 40) or [40, 48) will be determined. In the latter case, the new quantized value becomes 40 sgn (x_n), and in the former case, the quantized value remains 32 sgn (x_n). The refinement pass adds another bit of precision in the binary expansions of the scaled transform values $\{x_k/(2T_0)\}$.

Following the refinement pass, the new scan order part is performed. This is the part of the Aswdr algorithm where ideas similar to the Taws selection principles are employed. A new scan order is created by a bootstrap process proceeding from higher to lower levels of the wavelet transform. This bootstrap process is called the Aswdr *new scan order procedure:*

At the highest level (which contains the trend coefficients), use the indices of the remaining insignificant values as the scan order at that level. Assuming that the new scan order is already created at level r, a new scan order is created at level $r-1$ in the following way. Use the old scan order to scan through the significant wavelet coefficients at level r in the transform. The first part of the new scan order at level $r-1$ contains the insignificant children of these significant wavelet coefficients. Rescan through the insignificant wavelet coefficients at level r. The second part of the new scan order at level $r-1$ contains the insignificant children, *at least one of whose siblings is significant*, of these insignificant wavelet coefficients. Rescan a second time through the insignificant wavelet coefficients at level r. The third part of the scan order at level $r-1$ contains the insignificant children, *none of whose siblings are significant*, of these insignificant wavelet coefficients. (Although this description is phrased in terms of a three-scan process, it can be performed in one scan by linking three separate chains at the end of one scan.) Use this new scanning order for level $r-1$ to create the new scanning order for level $r-2$, until all levels are exhausted.

The rationale for the creation of a new scan order is to reduce the number of steps between the new significant values which emerge when the threshold T is reduced to $T/2$. For example, let us consider Figures 9b and 9c. If the value of the threshold T is 24 for the Peppers image, then the first part of the new scan order for the transform coefficients in Figure 9c are the insignificant children of the significant parent locations shown as white pixels in Figure 9b. This captures a high percentage of new significant children within the first part of the new scan order, which thus greatly reduces the number of steps and hence the number of bits needed for encoding. Notice also how the locations of significant values are highly correlated with the locations of edges within the Peppers image. The scanning order of Aswdr dynamically adapts to the locations of these edge details in an image, and this enhances the resolution of these images in Aswdr-Compressed images.* Because the Taws algorithm makes use of a similar dynamically adapted scanning order, it also enjoys high resolution of edges. Consequently, as seen in Section VI, Taws-denoised images are sharper, more in focus, than denoisings obtained with other wavelet methods.

The Aswdr procedure continues either until the threshold T is less than some preassigned value 𝟙, or until a preassigned number of bits (a bit budget) has been used for encoding. The first stopping criterion is used by the Taws algorithm, whereas the second stopping criterion is used by the Taws-Comp simultaneous compression and denoising algorithm.

> Read full chapter

Topics in Multivariate Approximation and Interpolation

Tomas Sauer, in Studies in Computational Mathematics, 2006

5.4 What remains is interpolation

Proposition 14 states that for a given grading and a given inner product (which can be seen as a part of the grading), there is a *unique* normal form $\nu_{□}(f)$ for a polynomial f modulo an ideal □, and the polynomial $\nu_{□}(f)$ is the canonical representer of the equivalence class $[f] = f +$ □ in $\Pi/$□. This finally returns us to interpolation as it says that normal forms are natural interpolants for ideal interpolation schemes.- **Theorem 15***If* □ □ Π□ *has the property that* □(□) := ker □ *is an ideal in* Π, *then is a degree reducing interpolation space for* □ and the interpolation operator takes the form .*Proof.* Since det P_{00}(□$_0$) ≠ 0, the mapping $\nu_{□}$(□) is an interpolation operator whose linearity follows from the uniqueness of remainders and the fact that $f + f$□ has the □–representationDegree reduction of the normal form operator, on the other hand, is seen directly from the reduction algorithm.

Hence, normal forms are canonical degree reducing interpolants and is the canonical degree reducing interpolation space. Moreover, reduction even allows us to compute an interpolant to a polynomial f if the interpolation problem is only known in a *implicit* way, namely in terms of a basis of □(□). On the other hand, in the case of a Lagrange interpolation problem, that is, if a finite node set is given, we can easily give the Lagrange fundamental polynomials *explicitly*. To that end, let be any vector such that νT(□ – □□) ≠ 0, □, □□ □ □ – such vectors exist in abundance, in fact there are only finitely many (normalized) vectors that do not have this property. Now simply define for □ □ □ the polynomials

(29)

and ℓ□, □ □ □, is the Lagrange basis of , so that the interpolant takes the form

But this tells us even more: the basis is obtained by taking the normal forms of polynomials of total degree #□ − 1, leading to the following observation.**Corollary 16***For any finite node set the normal form interpolation space satisfies* .In other words, the minimal degree interpolation space is spanned by the polynomials $p□ = \nu_{□(□)}((\cdot)□)$ and the finite matrixhas the maximal rank #□. In contrast to the approach in [9], where infinite matrices are considered, the linear system associated to a Lagrange interpolation problem can therefore always be assumed to be finite, even if the number of polynomials exceeds the number of points significantly.Normal forms are closely tied to the *Hilbert function* of the ideal □(□). Though the Hilbert function of an ideal is usually defined and considered for the homogeneous grading, cf. [34,54,55], it can be extended in quite a straightforward way to arbitrary gradings, which we will describe next. For an ideal □ □ Π we set $W□(□) = W□(G)$, G any □–basis of □, and define the *homogeneous* Hilbert function of □ as(30)Moreover, the “summarized” functionis called the *affine* Hilbert function of □. The latter one is monotonically increasing, i.e., $H_□(□) \le H_□(□□)$ if □ ≤ □□. Also note that, strictly speaking the second identity in (30) is only valid if $\Pi□_0$ is of finite dimension which is the case, for example, for the grading by total degree, but not for the one from (22). The affine Hilbert function describes the dimensions of the nested interpolation spaces , $N_0 := P_{00}(□_0)_{-1}P_{00}$.**Proposition 17** *For □ □ □ we have that*(31)*Proof*. We first note that is spanned by homogeneous polynomials: any polynomial f □ $W□(□)$ □ $\Pi□_0$ has the property that $f = \nu_{□(□)}(f)$. Hence . On the other hand, any homogeneous p □ p^* □ $\Pi_{□0}$ also satisfies $p = \nu_{□(□)}(p)$ □ $W□(□)$ and therefore as well.

The understanding obtained so far of degree reducing interpolation for graded rings and the prominent role that normal forms play, finally allows us to define the notion of a *Newton basis* for arbitrary gradings, and even to construct such bases. To that end, we start off with the *homogeneous bases* , , which are defined by the requirement that

(32)

Since there is a finite set □* □ □ such that $P□_0$, □ □ □*, still form a homogeneous basis for . Moreover, we can assume that 0 □ □* as $W_0(□(□)) = \{0\}$ implies that $V_0(□(□)) = \Pi_{00}$ and thus (□(□)) = Π, i.e., .

The construction of the Newton basis proceeds as follows: beginning with 0 = min □□ and taking into account that the $P_{00}(□)$ has rank $H_{□(□)}(0)$, we can find $□_0$ □ □ such that det $P_{00}(□_0) \neq 0$, and so $N_0 := P_{00}(□_0)_{-1}P_{00}$ satisfies $N_0(□_0) = I$.

Suppose now that for the k smallest elements $0 = □_0 < □ < □_{k-1}$ of □□ we have constructed such that

and set $□ = □k = \min\{□□ □ □ : □□ > □_{k-1}\}$. Then the polynomials in the vector are linearly independent, vanish on k_{-1}, and their leading terms (P) span W(()). In addition, the rank of the matrix P(\$k_{-1}$) coincides with the cardinality of P. Consequently, there is a subset of \$k_{-1}$ such that det P() ≠ 0, and once we define the polynomial vector N = P()$_{-1}$$P$ it follows that N() = I. This process, first used in [16] in the case of monomial gradings based on term orders, is nothing but the –equivalent of *Gauß elimination by segments*, see [6], or the Gram–Schmidt process in [66], and creates a decomposition of into and a Newton basis N,   *, such that

Once more the normal form space serves as a prototype for degree reducing interpolation spaces in the sense that we can even *characterize* degree reducing interpolation in terms of the existence of a Newton basis. For the grading by total degree, this result has been given in [66], the presentation here, however, is strongly influenced by discussions with Carl de Boor on [67] and his comments made in [9].**Theorem 18***A polynomial subspace is a degree reducing interpolation space for a finite node set if and only if there exists a* Newton basis ,   * , *and a decomposition of into ,   * such that*(33)*Proof.* Suppose that is a degree reducing interpolation space. Together with and Theorem 15 this implies for any f Π thati.e.,(34)Let N,   *, denote a Newton basis of , constructed as above, and set . The interpolation property of then yields thatHence, for any   there exist N N and N N such that N – N () which is improved by (34) intoSince Π = span (N) + V(()) for all   , this implies (33).Conversely, suppose that has a basis that satisfies (33), hence, is an interpolation space. Moreover, the sets , defined by N = I()(N) also satisfy , and thus form a Newton basis for the normal form interpolation space . Now, for f Π,and since the leading terms of both sets, N and N, each span Π jointly with V(()), they consist of H()0() linearly independent elements, which leads to the conclusion that ; in other words, is a degree reducing interpolation space.

From the proof of Theorem 18 we can draw another interesting conclusion which further justifies the notation "*".**Corollary 19***Let   Kd. Then the index set * for the Newton basis is the same for all degree reducing interpolation spaces and takes the form*Allright, the existence of a Newton basis characterizes degree reducing interpolation spaces and of course those will depend on the parameters involved with the grading, that is, the monoid with its total order and the inner product that is used for the orthogonal projection. We should first note that the normal form interpolant is singled out by requiring that its Newton basis makes the decomposition in (33) an *orthogonal* one. But even then there is in general a freedom of choice as different decompositions of into ,   *, lead to different bases. This is, however, not a multivariate phenomenon as even the univariate Newton polynomialsdepend on the ordering of the points in which need not be in any "natural", for example ascending, order.It is clear that if N is a Newton basis for an interpolation space , then so isthat is, even after fixing the decomposition 𝒫𝓃, the elements of a Newton basis are only defined up to addition of an ideal element of at most the same degree – and this is a truly multivariate feature. On the other hand, the above modification is also the only way to pass from one Newton basis to another so that they are normal forms up to addition of ideal elements of restricted degree.**Corollary 20**If 𝛿() < min {𝛿(g) : g ∈ G}*for some* 𝛿*–basis G of* (𝒳) *then the degree reducing interpolation space is unique and so is its Newton basis up to indexing of nodes.*

> Read full chapter