Precision Improvement

of Software Algorithms

AEROLIS

– Methods of Analysis

– Solutions

Study done by André LIEUTIER of SUN Microsystems France for Aerospacial Toulouse Airpliane Division, development service CAO.

A. LIEUTIER

Translation by Mona Alemi

Summary

Appendices

1. Introduction

The **Aerolis**, developed by aerospatial, achieves the conception of the complex left surfaces. These surfaces are defined by polynomials of relatively high degree (up to twenty). Some operations temporarily go through representation of surfaces by higher degree polynomials.

The mathematical algorithms that are actually used for some operations require the calculations to be done with 30 significant digits so that the final result is within an acceptable precision (error less than a micron for global results of many meters).

Actually, these calculation are executed on computers having floating point coprocessors, where numbers are represented with 128 bits, that is almost 110 bits for the mantissa, which is more than 30 significant digits (especially CDC 930 and CDC 990).

Today, there is no workstation that is capable of fast floating point calculations on 128 bits. Only the MicroVax family of computers provide 128 bit calculations but because they are implemented in software response times are too large for an interactive software.

To be able to use the software on Sun workstations the first solution considered was to develop an arithmetic coprocessor in 128 bits and its interface to Sun Fortran. Beside the cost, this solution has evolution problems (must redesign board for performance enhancement), and portability problems. Especially such a solution is totally contrary to Sun's "Open System" philosophy, to A.B.I programs and Sparc. The quadruple precision type was not retained in Sparc specifications.

The other solution consists of modifying the algorithms used in **Aerolis** to get final results that are as precise as the intermediary calculations done on the standard Sun coprocessors where numbers are represented in 64 bits (double precision). The object of this study is to determine the feasibility of this last solution.

In order to simplify the analysis, the first objective was to execute with the necessary precision a Benchmark selected by Aerospatial as representative of the most critical precision problems. The results obtained with the modified algorithms on Sun (excellent precision and performance) helped to respond positively to the feasibilty question.

This note describes methods of analysis and the solutions.

Chapters 2 and 3 are general introductions on methodology of analysis and the notion of "unsuitable representaion" or "ill conditioning".

Chapter 4 describes the path to the actual solution in the special case of Reducs and Spctra.

Then there are Appendices which represent the mathematical developments used in programs or general ideas which may be applicable to **Aerolis**.

The only reference that was useful is the article "family of orthogonal polynomials" from Encyclopaedia Universalis which I have reproduced in the appendix.

All the developments and proofs presented have been done by myself.

This study wouldn't have been possible without the support and assistance of Frederic Andre and Patrick Jubert of Aerospecial.

This note is designed for the lecturers who know the Aerolis software.

2. Methods of localisation of critical points

The **Aerolis** software has 150,000 lines of source. Therefore we need a method for localizing critical points, that is zones where a source modification is necessary to have the desired precision in 64 bits.

This method was applied in the case of the Benchmark (approximately 3000 lines).

The first phase is obvious: we limit the analysis to modules where the CDC version uses "*double*" precision. (*double CDC* = 128 bits and *single CDC* = 64 bits = *double sun*).

In the case of the Bench, more than half the modules use the option "double CDC". So on Sun, those different modules will contribute to the total error. Things are more complex in reality but in order to explain the method we can simplify: if modules A and B respectively bring errors of 1 and 1000 on the final result, we could not test the improvements brought by the module A as long as we haven't made module B precise. The inverse however is not true.

The second phase consists of executing the benchmark on the CDC with all the modules in double except one each time. The module that brings the highest final error will be the first to study. We could accelerate the method by proceeding dichotomically: each half of the modules is executed without the "double", then each fourth, etc.

That's how we found in the case of Benchmark that the module reducs is responsible for the biggest part of the error. So I had to start the correction with it. It's only after I have corrected it that I have found a secondary source of error, negligable compared to the one in reducs but still too large to be acceptable.

The third phase of localization of the critical points is similar to the second but more subtle. We don't look for which module but which data representation is responsible for the error. This consists of successively truncating different double tables to single ones (simply by converting them to single and again to double: that way we have lost all the significant digits after the 16 or the 17th one) and to see the effect on the final result. If the calculations done in double precision

on this truncated table gives wrong results then we should have a data representation that conditions better the final result.

This third method raises an essential point: the method of analysis of the conditioning of the algorithms. It is more useful to state the problems in terms of conditioning of the intermediate results rather than the loss of precision in different treatments. For example, if matrix A has small pivots when inverting it and therefore has an inverse with large coefficients, it's useless to improve the matrix inversion: we need simply to change the matrix. If the problem is physically well stated (good conditioning) then there should exist a method that will not go through ill conditioned representations.

It's this intuition that has guided my work.

3. The notion of "bad" representation of data.

For the purpose of clarification I will illustrate what I mean by bad data representation in a simple case.

Let's consider in $\mathbb{R}^2$ the base $(1,0)$ et $(1,\varepsilon)$ with $\varepsilon = 10^{-20}$. If we want to represent the point $(1,-1)$ in this base, we would write :

$$(1,-1) = (1+10^{20}) \times (1,0) + (-10^{20}) \times (1,\varepsilon)$$

The new coordinates of the point is $(1+10^{20})$ and (-10^{20}). If we use these new coordinates on a machine that like Sun works almost with 17 significant digits $(1+10^{20})$ is impossible to represent without "losing the one". The only fact of transforming the coordinates of $(1,-1)$ in the new basis and to put them back in the old basis gives errors larger than the order of magnitude of the initial coordinates.

Representing the points of the plane in this basis is a typically bad representation of data (compared to the canonical basis of $\mathbb{R}^2$).

The fundamental problem of Aerolis is that the representation of polynomials of high degree in the canonical basis $1, x^1, x^2, x^3, \ldots, x^n$ is a bad representation of data compared to the defined geometry. The representation in the Bernstein basis is much better since it is closer to the defined geometry. All the conceived solutions are changes of the representation basis of the polynomials, better than the representation in the canonical basis. The exact definition of what a good representation is can be the objective of a future note.

4. Reducs and Spctra

4.1. The purpose of Reducs

Given a polynomial P of degree N, find the nearest polynomial Q of degree n in the sense of the norm L_2 on $[0,1]$, with of course $n < N$.

P is defined by its coefficients in the canonical basis, and we are looking for Q also under this form. We note: $< f \,.\, y >= \int_0^1 f(x) \,.\, y(x) \,.\, dx$ the scalar product L_2.

P is defined by :

$$P(x) = \sum_{i=0}^{N} P_i \,.\, x^i$$

Q is defined by :

$$Q(x) = \sum_{i=0}^{n} Q_i \,.\, x^i$$

• The problem can be written:

Q minimum of $< (P-Q) \,.\, (P-Q) >$ with $Q \in \Pi_n$ (Π_n sub-space of polynimials of degree$\leq n$, Π_n of dimension $n+1$).

• Another way of formulating the problem and that leads naturally to the solution that I considered can be stated :

Q is the orthogonal projection of P on Π_n.

Using the classical method of "least squares", the differential of the quadratic form brings a linear system:

Q minimum of $< (Q-P).(Q-P) >$.

$$
\begin{array}{ll}
\Leftrightarrow \frac{\partial}{\partial Q_i}\left[< (Q-P).(Q-P) > \right] = 0 & \forall i = 0,1,2,\ldots,n \\
\Leftrightarrow 2 < \frac{\partial}{\partial Q_i}(Q-P).(Q-P) >= 0 & \forall i = 0,1,2,\ldots,n \\
\Leftrightarrow < x^i.(Q-P) >= 0 & \forall i = 0,1,2,\ldots,n \\
\Leftrightarrow < x^i.Q >=< x^i.P > & \forall i = 0,1,2,\ldots,n \\
\Leftrightarrow < x^i.\sum_{j=0}^{n} Q_j.x^j >=< x^i.P > & \forall i = 0,1,2,\ldots,n \\
\Leftrightarrow \sum_{j=0}^{n} < x^i.x^j > .Q_j =< x^i.P > & \forall i = 0,1,2,\ldots,n
\end{array}
$$

It's the linear system for which the matrix is $< x^i.x^j >$ and the second member $< x^i.P >$. We have :

$$< x^i.x^j >= \int_0^1 x^{i+j} = \frac{1}{i+j+1}$$

The inverse of the matrix $< x^i.x^j >$ can be written in an analytical form which would be an improvement as we will avoid inherant errors to a matrix inversion algorithm. Unfortunately, even without any error in the inversion, this method goes through intermediate values which are "ill representations".

In fact, in the case $n = 15$, the inverse matrix of $< x^i.x^j >$ has terms of order 10^{20}.

This means that a variation ε on one of the second members can cause a variation of $10^{20}.\varepsilon$ on the solutions Q_i. So, even if the second members are calculated without error,the relative rounding errors of 10^{-17} by themselves can completely invalidate the final result.

It is therefore useless to look further in this sense: any method using scalar products $< x^i.P >$ as intermediate representation of P is unusable in Sun double precision.

4.3. The first proposed method

All the methods consist of representing the polynomials in a basis other than the canonical basis.

The primary problem of the canonical basis being that the inverse ($< x^i . x^j >$) has very large terms, we will look for a basis b_i such that the inverse of $< b^i . b^j >$ has terms of a smaller order of magnitude.

I have chosen the normalized Legendre polynomial basis which verifies:[1]

$$\left(< L^i . L^j > \right) = (\delta_{ij}) = \left(< L^i . L^j > \right)^{-1}$$

and that has the advantage of being a more elegant solution to the stated problem

The $(L^0, \ldots, L^n)$ form an orthonormal basis of Π_n. The coordinates of the orthogonal projection of P on Π_n are the respective scalar products of P with the basis vectors:

$$Q = \sum_{i=0}^{n} < L^i . P > . L^i$$

We have:

$$< L^i . P > = < L^i . \sum_{j=0}^{N} P_j . x^j > = \sum_{j=0}^{N} P_j . < L^i . x^j >$$

In the actual method, the $< L^i . x^j >$, for $i = 0, 1, 2, \ldots, 20$ and $j = 0, 1, 2, \ldots, 80$ are precalculated on the CDC and read as a table of values when the program is initialized. In fact, with the second method, this calculation can be done on Sun while preserving an acceptable precision. Another solution will consist in calculating $< L^i . P >$ by numerical integration of the Gaussian method (see appendix 2).

[1]*see Annexe 1

4.4. The second proposed method

The old method gives the correct results until $n = 8$. The first method is valid for n much larger, but it is impossible to approximate P to less than 10^{-1} ou 10^{-2}. In fact, if in the version CDC quadruple precision, we truncate the table defining the polynomial P (in the canonical basis), the final result is wrong with 10% error. The problem is already badly stated in Reducs because the knowledge of the coefficients of P in double precision ill conditions the final result. Only a global solution Spctra/Reducs can give a satisfactory precision.

The polynomial P is the result of composition of two polynomials. The calculation of the coefficients of P is done in Spctra. As soon as these coefficients are calculated, the information is lost, we don't have any more chance to calculate Q. So we have to represent P under a better form than its coordinates in the canonical basis. For this there are a number of solutions. The values of P for the $\frac{n+N}{2}$ points of Gaussian interpretation seem to be an elegant representation, (see appendix 1) but I have choosen the representation easiest to implement in the actual code and that has the advantage, if we generalize it, to be able to solve other precision problems. We represent P in the basis $\left(x-\frac{1}{2}\right)^n$ which can be expressed with a translation on the parameter:

$$u = x - \frac{1}{2} \quad , \quad u \in \left[-\frac{1}{2}, \frac{1}{2}\right]$$

with substituing $\left(u+\frac{1}{2}\right)$ for x e obtain a polynomial that describes the same curve with a parameter that doesn't go over $[0,1]$ but $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

Practically, we have : $P = f \circ y$.

We substitute $\left(u+\frac{1}{2}\right)$ in the definition of y, we use the same algorithm of calculation of composite polynomial's coefficients and we obtain the coefficients of P in the new basis.

We certainly need to change Reducs ; we have this time :

$$< L^i . P >= \sum_{j=0}^{N} P'_j . < L^i . \left(x - \frac{1}{2}\right)^j >$$

the $< L^i . \left(x - \frac{1}{2}\right)^j >$ are calculated with integration on the variable $u = x - \frac{1}{2}$, with the analytical calculation of the coefficients of the Legendre polynomials on $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

The canonical basis on $\left[-\frac{1}{2}, \frac{1}{2}\right]$ is a better representation compared to the norm L_2 than the canonical basis on $[0, 1]$. The rigorous explanation of this fact is very technical and can be done in a future note if necessary.

ANNEXE 1

Family of orthonormal polynomials

1) Définition

Given $[\alpha, \beta]$ an interval of $\mathbb{R}$. We choose a given interval because we will need further on to calculate the family of orthonormal polynomials on $[0,1]$ and on $\left[-\frac{1}{2}, \frac{1}{2}\right]$.

We note Π_n the vector space of dimension $n+1$ formed by polynomials of dimension $\leq n$; Π_n^0 vector space of dimension $n-1$ of polynomials Π_n with roots in α et β. Given :

$$a = (x-\alpha)(x-\beta)$$

notice that any non zero element of Π_n^0 factorizes a.

We will establish the formulas that lets us compute the coefficients of the family of the polynomials that verify

1. Legendre :

$$\left|\begin{array}{l} \text{for } n \geq 0 \\ L_n \text{ of degree } n \\ \text{and } \forall f \in \Pi_n \,,\ \int_\alpha^\beta f(x) \,.\, L_{n+1}(x)\, dx = 0 \end{array}\right. \tag{1}$$

2. (No name that I know of) :

$$\left|\begin{array}{l} \text{for } n \geq 2 \\ S_n \text{ of degree } n \,,\ S_n \in \Pi_0^n \\ \text{and } \forall f \in \Pi_n^0 \,,\ \int_\alpha^\beta f(x) \,.\, S_{n+1}(x) \,.\, dx = 0 \end{array}\right. \tag{2}$$

we can easily verify that these definitions are equivalent to :

1.' $\{L_0, L_1, L_2, \ldots, L_n\}$ orthogonal basis of Π_n ; $\forall n \geq 0$.

2.' $\{S_2, S_3, S_4, \ldots, S_n\}$ orthogonal basis of Π_n^0 ; $\forall n \geq 2$.

orthogonal is relative to scalar product L_2 :

$$< f \,.\, g >= \int_\alpha^\beta f(x) \,.\, g(x)\, dx$$

Note that these definitions are sufficient to determine uniquely L_n and S_n within a multiplier coefficient ; In fact, the first definition can be expressed : "L_n is in the sub-vector space othogonal to Π_{n-1} in Π_n" ; which is of dimension 1. Same reasoning for S_n.

2) Usefulness

The L_n are used in Reducs to find the orthogonal projection of the polynomial P on Π_n. In Reducs, we use the coefficients of L_n on $\left[-\frac{1}{2}, \frac{1}{2}\right]$ to calculate the scalar products with the "translated" polynomial and the coefficients of L_n on $[0, 1]$ to express the solution Q in the canonical basis.

The S_n answer to the same problems but in the case that we have contraints in the extremes of the interval. Find the polynomial Q of degree n nearest in the sense of L^2 to a polynomial P of degree N and proving :

$$Q(\alpha) = P(\alpha) \quad \text{et} \quad Q(\beta) = P(\beta)$$

We have :

$$P'(x) = P(x) - P(\alpha) - \left[P(\beta) - P(\beta)\right] \frac{x - \alpha}{\beta - \alpha}$$

we have: P' de degré N et $P'(\alpha) = P'(\beta) = 0$.

We project P' on Π_n^0 which will give a polynomial Q' and the solution Q can be expressed:

$$Q(x) = Q'(x) + P(\alpha) + \left[P(\beta) - P(\alpha)\right] \frac{x - \alpha}{\beta - \alpha}$$

3) Method of calculation

We can derive from the Rodrigues formula the following :

$$\bullet\ L_n = D^n[a^n] \tag{3}$$

$$\bullet\ S_n = \frac{1}{a} D^{n-2}[a^n] \tag{4}$$

where D^n means the nth derivative.

It is easy to prove that these formulas verify the respective definitions (1) et (2). Let's do it for the first : Given P a polynomial of degree $n-1$.

$$\begin{aligned}\int_\alpha^\beta L_n . P &= \int_\alpha^\beta D^n(a^n) . P . dx \\ &= \left[D^{n-1}(u^n) . P\right]_\alpha^\beta - \int_\alpha^\beta D^{n-1}(a) . P' \, dx\end{aligned}$$

(integration by parts).

But $a^n = \left[(x-\alpha)(x-\beta)\right]^n$ has in α and β poles of order n so the derivative of a^n until order $n-1$ become zero at α et β. The term in brackets is zero so. We reiterate n n time the operation, which will give :

$$\int_\alpha^\beta L_n . P = (-1)^n \int_\alpha^\beta a . D^n(P) . dx = 0$$

because P is of degree $n-1$. The proof of S_n is similar.

Therefore the formulas (3) et (4) give the orthogonal family. It's more practical to use the orthonormal family. Let's clarify the normalization in the case of L_n :

$$\tilde{L}_n = \frac{L_n}{\sqrt{< L_n . L_n >}} \quad < L_n . L_n >= \int_\alpha^\beta D^n(a^n) . D^n(a^n) . dx$$

like the proof for (3), we do n integrations in parts :

$$< L_n . L_n >= (-1)^n \int_\alpha^\beta a^n . D^{2n}(a^n) . dx$$

a^n is a polynomial of degree $2n$ that has a coefficient of the highest term 1 so:

$$D^{2n}(a^n) = D^{2n}(x^{2n}) = (2n)!$$

and

$$< L_n \,.\, L_n > \;=\; (-1)^n (2n)! \int_\alpha^\beta a^n \,.\, dx$$

$$< L_n \,.\, L_n > \;=\; (-1)^n \,.\, (2n)! \int_\alpha^\beta (x-\alpha)^n (x-\beta)^n \, dx$$

again using integration by parts :

$$= (-1)^n (2n)! \Big[\Big[\tfrac{1}{n+1} (x-\alpha)^{n+1} \,.\, (x-\beta)^n \Big]_\alpha^\beta - \int_\alpha^\beta \tfrac{n}{n+1} (x-\alpha)^{n+1} (x-\beta)^{n-1} \, dx \Big]$$
$$= (-1)^{n+1} (2n)! \int_\alpha^\beta \tfrac{n}{n+1} (x-\alpha)^{n+1} (x-\beta)^{n-1} \, dx$$

we reiterate n times the integration by parts:

$$< L_n \,.\, L_n > \;=\; (-1)^{2n} (2n)! \, \frac{(n!)^2}{(2n)!} \int_\alpha^\beta (x-\alpha)^{2n} \,.\, dx$$

$$< L_n \,.\, L_n > \;=\; (n!)^2 \, \frac{(\beta-\alpha)^{2n+1}}{2n+1}$$

4) Development

In order to show that these formulas lead to an efficient and precise method of calculation of the coefficients L_n and S_n, we develop the calculations in the particular case of L_n with $\alpha = 0$ and $\beta = 1$:

$$\begin{aligned} L_n &= D^n(a^n) = D^n\Big[x(x-1)\Big]^n = D^n \sum_{i=0}^{n} C_n^i (-1)^{(n-i)} . x^{n+i} \\ &= \sum_{i=0}^{n} C_n^i (-1)^{(n-i)} \frac{(n+i)!}{i!} x^i \\ L^n &= \sum_{i=0}^{n} l_i^n . x^i \quad \text{with} \quad l_i^n = (-1)^{n-i} \frac{n!(n+i)!}{(i!)^2(n-i)!} \end{aligned}$$

and if we normalize:

$$\sqrt{< L_n . L_n >} = \sqrt{(n!)^2 \frac{1}{2n+1}} = \frac{n!}{\sqrt{2n+1}}$$

$$\tilde{L}^n = \sum_{i=0}^{n} \tilde{l}_i^n . x_i \text{ avec } \tilde{l}_i^n = (-1)^{n-i} . \sqrt{2n+1} \frac{(n+i)!}{(i!)^2(n-i)!}$$

The method of computation of these coefficients is then iterative:

$$\left[\begin{array}{l} \tilde{l}_0^n = \sqrt{2n+1} \\ \tilde{l}_{i+1}^n = \tilde{l}_i^n \times \frac{-(n+i+1)(n-i)}{(i+1)^2} \end{array} \right. \tag{5}$$

(we can invert the sign).

This method is more costly and very precise because we don't do multiplications and divisions.

APPENDIX 2

Integration by the Gaussian Method

Eventhough this technique is not used in the actual version of the benchmark, I will present it here because it leads to another type of representation of a polynomial : the knowledge of its values in n points.

We go back to the notation of appendix 1. Eventhough these techniques exist in certain books, I don't know of any reference that talks about precision problems, probably because, used in finite elements, we restrict ourselve to weak degrees. That's why we integraly reproduce my reasoning about the superiority of the Gaussian method in precision.

1. Integration methods of Newton-Cotes

Given P a polynomial of degree less than or equal to $N-1$. Soient $x_1, x_2, \ldots, x_N$, N distinct points of $[\alpha, \beta]$. The values of $P(x_1)$, $P(x_2), \ldots, P(x_N)$ detemine P uniquely, more precisely :

$$\varphi : \Pi_{N-1} \mapsto \mathbb{R}^N P \mapsto \Big(P(x_i), i = 1, \ldots, N\Big)$$

φ is an isomorphism of the vector space (bijective linear application). In fact, it is evident that φ is linear and that the kernel of φ is reduced to 0.

If we note S the linear form integrated on $[\alpha, \beta]$" :

$$S(P) = \int_{\alpha}^{\beta} P(x)\,.\,dx$$

We have :

$$\begin{array}{ccc} \Pi_{N-1} & \xrightarrow{\varphi} & \mathbb{R}^N \\ \Big\downarrow S & & \\ \mathbb{R} & & S \circ \varphi^{-1} \end{array}$$

$S \circ \varphi^{-1}$ is a linear form on $\mathbb{R}^N$ and can be represented by an N uplet (tenseur covariant) :

$$S \circ \varphi^{-1} \left(\begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{pmatrix} \right) = (\alpha_1, \ldots, \alpha_N) \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$$

which can be written :

$$S(P) = \int_\alpha^\beta P(x) \, . \, dx = \sum_{i=1}^{N} \alpha_i \, . \, x_i$$

This is the Newton-Cotes integration method.

The α_i can be simply calculated : Given $B_i \in \Pi_{N-1}$, defined by :

$$B_i = \frac{\prod_{\substack{j=1 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{N} (x_i - x_j)}$$

We have :

$$B_i(x_j) = \delta_{ij}$$

and

$$\int_\alpha^\beta B_i(x) \, . \, dx = \sum_{j=1}^{N} \alpha_j \, B_i(x_j) = \alpha_i$$

Given $U \in \Pi_{N-1}$ the polynomial defined by :

$$U(x) = 1, \quad \forall x \in \mathbb{R}$$

We have :

$$\int_\alpha^\beta U(x) \, . \, dx = (\beta - \alpha) = \sum_{i=1}^{N} \alpha_i$$

The sum of α_i is $\beta - \alpha$ but, in the case that x_i are equidistant, the x_i have alternate signs and very large absolute values , that is why this method is imprecise for a high degree.

2. Method of Gaussian integration

Consider Π_{2N-1}. Given P a polynomial of degree inferior or equal to $2N-1$. Given L_N a polynomial of degree N on $[\alpha,\beta]$. We know that there are two unique polynomials of degree inferior or equal to $N-1$ such that :

$$P = L_N \,.\, Q + R$$

Q and R are respectively the quotient and the remainder of the divion of P by L_N.

The theorem on the orthogonal polynomials (see article) says that L_N has N real distinct roots in $]\alpha,\beta[$. Given $x_1, x_2, \ldots, x_N$ these valuse, we have:

$$\forall\, i = 1,2,\ldots,N, \quad P(x_i) = L_N(x_i)\,.\,Q(x_i) + R(x_i) = R(x_i)$$

On the other hand :

$$\int_\alpha^\beta P(x_i)\,dx = \int_\alpha^\beta L_N(x)\,.\,Q(x)\,.\,dx + \int_\alpha^\beta R(x)\,.\,dx$$

By definition of L_N, the first term of the second member is zero and :

$$\int_\alpha^\beta P(x)\,dx = \int_\alpha^\beta R(x)\,.\,dx$$

we could apply the NewtonCotes method to R, polynomial of degree less than or equal to $N-1$: $\exists\ \alpha_1,\ldots,\alpha_N$ such that :

$$\int_\alpha^\beta P(x)\,.\,dx = \int_\alpha^\beta R(x)\,.\,dx = \sum_{i=1}^{N} \alpha_i\,.\,R(x_i) = \sum_{i=1}^{N} \alpha_i\,.\,P(x_i)$$

the first advantage of this method is that it is enough to know that value of a polynomial of degree $2N-1$ in N points to be able to compute its integral. But the first advantage of this point of vue is precision. In fact, this time we can compute differently the coefficients

α_i : we have:

$$B_i(x) = \left[\frac{\prod_{\substack{j=1 \\ j \neq i}}^{j=N} (x - x_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{j=N} (x_i - x_j)} \right]^2$$

We have : $B_i(x_j) = \delta_{ij}$ et B_i de degré $2N - 2$, so:

$$\int_\alpha^\beta B_i(x)\, dx = \sum_{j=1}^{N} \alpha_j \,.\, B_i(x_j) = \alpha_i \,, \quad \text{or} \quad B_i(x) \geq 0 \,, \forall x \in \mathbb{R}$$

therefore:

$$\alpha_i > 0 \,, \forall i = 1, 2, \ldots, N \quad \text{et} \quad \sum_{i=1}^{N} \alpha_i = \beta - \alpha$$

This time, the α_i are all positive of teh same order of magnitude than $\frac{\beta - \alpha}{N}$, therfore a better precision.

3. Application to Spctra/Reducs

Given N the degree of the polynomial P to be approximated and $n$the maximum degree of the desired polynomial (here $n = 19$).

The method described in chapter 4 requires the calculation of the scalar products:

$$< L_i \,.\, P >= \int_\alpha^\beta P(x) \,.\, L^i(x)\, dx \quad \text{with} \quad i \leq n$$

It's enough to know $P(x_j)$ for $\frac{n+N}{2} + 1 = K$ points x_j root of L^K to calculate precisely $< L^i \,.\, P >$ by the method of Gaussian integration. The idea is that $P(x_j), j = 1, \ldots, K$ define a very good representation of P that can eventually replace the decomposition in the basis of $\left(x - \frac{1}{2}\right)^i$.