Computers Math. Applic. Vol. 26, No. 11, pp. 3–135, 1993
Printed in Great Britain.

0898-1221/93 $6.00 + 0.00

Convolution Number

W. C. HASSENPFLUG
Department of Mechanical Engineering, University of Stellenbosch
Stellenbosch, South Africa

(Received March 1993; and accepted May 1993)

Abstract—*Convolution number* is a new proposed name for the sequence of numbers that constitute the coefficients of polynomials and truncated Taylor expansions of functions. The arithmetic of the convolution number is the well-known arithmetic of sequences, where multiplication is the convolution of sequences, or the arithmetic of polynomials or formal power series where the multiplication is the Cauchy product. To separate the coefficients from the Taylor series formally, a *Taylor transform* is defined. Considering the sequence of coefficients as a single number is a new perspective which emphasizes the purely computational application, making a (digital) computer method out of a symbolic (computer) method. By this means, the whole well-known field of solution by Taylor series is cast in a simple numerical application oriented algebraic method. Convolution number analysis is an alternative computational tool to obtain numerical solutions for certain problems that would otherwise be determined by analytical analysis, or Finite Difference or Finite Element methods when analytical solution is too difficult. Taylor expansion of functions of a single variable, i.e., univariate polynomials, real or complex, are considered to develop the theory and the methods. All the methods and results are adapted to functions of two variables, i.e., bivariate polynomials, from which the extension to multivariate polynomials is then obvious.

Simple programming of the four basic arithmetic operations on the convolution number is reviewed (and the square root operation), to be used as a set of subroutines, so that problems are formulated and programmed directly in terms of convolution numbers.

It is shown how matrix algebra using convolution numbers as elements can be applied to vibration problems and illustrated with an example, resulting in a dynamic system matrix, although limited to small degree of freedom systems because of the large storage required.

From the arithmetic, an algebra of convolution numbers is developed, considering the convolution number as a variable. An example of conformal mapping by convolution number algebra is given.

A convolution function of a convolution variable is defined, with a *compatibility condition* analog to the Cauchy-Riemann equations of a complex function of a complex variable. The compatibility equations serve as a tool to derive some basic theorems of convolution variables which are necessary for the development of programming with convolution variables.

Generally, the convolution number represents a truncated Taylor series of a function. The arithmetic is such that each coefficient has original machine accuracy and, therefore, the corresponding function could be evaluated to machine accuracy, although no theoretical general rule for the *a priori* required length of the convolution number and the radius of convergence is given.

The well-known problems of polynomial composition and reversion of series are stated in terms of convolution number algebra. It is shown, by an example, how such problems are solved on a digital computer once the basic arithmetic routines are programmed.

The well-known method to solve nonlinear ordinary differential equations by Taylor series is generalized to a simple computational solution of the integration process with the aid of a pointer in the computer storage of the convolution number. The solution process is posed in terms of a flow diagram, which is an exact copy of the analog computer diagram of a differential equation, from which the sequence of convolution number equations are programmed. Relation to the z-transform and digital filters is shown.

The same example of conformal mapping is solved as a nonlinear equation, and the solution of a nonlinear ordinary differential equation by convolution number analysis is presented.

In view of the applications, the radius of convergence of the function in the complex plane must be considered at all times, for which, unfortunately, no general theoretical determination can be given, and which may severely limit the advantage of a Taylor series solution. In the examples, it is shown how a practical estimate can be made. A final solution consists of the well-known method of patching

up a sequence of Taylor series, similar to a sequence of high order Finite Difference solution values; the difference, however, being that the series accuracy can be tested analytically.

The stability of the recursive solution routines is investigated.

The convolution number and its algebra is defined for a bivariate polynomial and an example of the solution of a nonlinear partial differential equation given. Again, the solution is seriously limited by the region of convergence.

Throughout, a distinctive and precise symbolic notation for convolution algebra has been attempted.

1. INTRODUCTION

Numerical results of problems of physics and engineering have been obtained by the contemporary available computational tools which have evolved during the course of time and which have influenced the development and preference of analytical or numerical methods.

Analytical methods have been used to develop the equations which are the problem statement, also called the governing equations. Analytical analysis was then used to proceed to the solution in analytical form. Numerical results were obtained by evaluating these final analytical expressions numerically with the increasing amount of developed aids up to the present digital computer. This method is still pursued presently as the preferred method. Let us call this the *analytical path.* The biggest disadvantage of analytical methods is that even slightly different problems have vastly different solutions, e.g., expansion into series of functions is only possible for a few selected simple boundaries, each one of these again require different special functions, of which large collections have been made up, e.g., [1,2]. Furthermore, when it comes to numeric evaluation, it is found that very few computer libraries have these special functions available.

Numerical methods, on the other hand, have been developed quite early to avoid difficult analysis and for cases for which analytical solutions could not be found. These methods have increased in importance with the arrival and improvement of the digital computer, and additionally, the Finite Element Method was specifically developed for the digital computer. Starting from the analytic equations of the problem statement the solution is obtained by numerical analysis. Let us call this the *numerical path.* The biggest advantage of numerical methods is that they are general, whole classes of problems are treated by the same methods.

The alternative method of numerical computation by graphical methods, nomographs and the analog computer has been ended by the digital computer.

Eventually analytic analysis has also benefited from the computer by the development of symbolic computer languages, [3,4]. The original perception of analytical results, i.e., that it indicates the character of the solution, has quickly been lost as symbolic computers generated analytic expressions of incomprehensible magnitude. This development has already been recognized and symbolic computer programs are specifically designed to generate efficient digital computer programs. Yet, although supported by computer, this is still the analytical path.

Analytical so-called closed form solutions are those which express the result in terms of known analytic functions. Conversely, if such a reference to known functions cannot be made, then it is considered that an analytic solution cannot be found. There is an inconsistency in this approach. The conversion of analytical functions to numerical values depends mainly on a Taylor series. Therefore, any result could be expressed directly in terms of a Taylor series within a certain region instead of using functions of functions which, in the end, have to be evaluated by a Taylor series anyway. A typical example is elliptic integrals which can all be expanded into a Taylor series within a certain region, yet a large analytical effort is made to reduce all to the three basic types [5]. This is a remainder of the pre-digital computer era when the effort of evaluation was only expended on the three basic types. To do this on a digital computer with a large effort to create efficient programs for only the three types [6], is totally inappropriate, as more time is wasted due to the elaborate expressions in terms of these three. The inconsistency now is that the known functions are selected rather arbitrarily because any new solution that can be expanded in a Taylor series could be defined as a known function.

Coefficients constitute the analytical-numeric interface, and analysis is really a manipulation of coefficients. In the particular case of the Taylor series, for practical applications, only the numerical values of the coefficients are necessary, and those only within machine accuracy rather than as formulas in terms of initial problem values, which, in practical applications, are only available in machine accuracy anyway.

Methods to generate Taylor coefficients have been perceived as analytical methods in the past. The purpose of this treatise is to show that with the concept of the coefficients as a single generalized number, it becomes a simple algebraic method that can be programmed as simple as algebra of real numbers, and that can be solved with a few general numeric subroutines. The Taylor coefficients are treated numerically from the beginning of the equations of the problem statement.

If the Taylor series of a function $a(x)$ is multiplied with a Taylor series of a function $b(x)$ to obtain the Taylor series of the algebraic product $a(x)b(x)$, then the product is called a Cauchy product, [7,8], referring to the method of collecting products of the coefficients of the two series and attaching the variable powers to them. The actual multiplication consists of manipulating the sequences which is the well-known convolution of sequences. In [9], it is also called serial multiplication. Defining the convolution as product has lead to the algebra of sequences [10], based on a similar algebra of functions by Mikusiński [11]. We go one step further by calling the sequence a number, so that the convolution of sequences become the product of convolution numbers. This is analog to the product of complex numbers, rather than the complex product of pairs of numbers. With the concept of a single number, we can pursue the algebra much further and develop an analysis, analog to analysis of complex numbers, and define convolution functions of a convolution variable, analog to complex functions of a complex variable.

For the purpose of a numerical algebra, we define the *convolution number* as the sequence of numbers that constitute the coefficients of the truncated Taylor series of a function.[1] By separating the coefficients from the variable terms and calling them a number, we obtain a purely numerical quantity, suitable for the digital computer, analog to the separation of coefficients of simultaneous linear equations into a new quantity called matrix. By this means, computation with polynomials becomes a purely numerical method which, with the development of a few arithmetic routines, can be used as easily as matrix algebra. Because we want to emphasize that the symbolic variables are not included as is customary in present symbolic algebra, we do not call this polynomial algebra, just like the algebra of linear equations is not called linear equation algebra but matrix algebra.[2]

Computation with polynomials or formal power series is well-known [7,8,12–14] and much development has been made to create efficient routines of its algebra, [15–17]. More recently, it was shown by Adomian, [18–22], how nonlinear algebraic and nonlinear ordinary differential equations, and in [19,20,23], partial differential equations can be solved systematically by using what is called the composition method, but it is a different approach and not numerically based as presented here.

By considering the sequence of coefficients as a single quantity like a number, written symbolically like a single number and in the computer with a single name, an analysis is developed which is as efficient as ordinary analysis of real numbers, the large sequence of numbers being handled internally in the digital computer, similar to the large number of digits that are handled in the computer in real number algebra, and even with a similar means of truncation. A similar idea for the Fourier spectrum of functions is described in [24], appropriately called harmonic variables.

[1]The German of convolution is Faltung. Therefore, the German translation of convolution number would be Faltungszahl.

[2]Unfortunately, the rather modern term *Linear Algebra* has confused this distinction again.

Such a method has the same characteristic as numerical methods, in that it is a single general method for all problems. As a result, we cannot recognize special properties of solutions, e.g., a simple solution like $\frac{1}{1-x}$ will appear as an infinite Taylor series.

With this computational tool, the path of analysis for problems that have solutions that can be expressed as a Taylor series is somewhere between the symbolic and the numeric, since this method could be termed semi-analytical, having the numerical coefficients of an analytic form of the solution.

For easy recognition in reading, we employ a special symbol for the convolution number, similar to special symbols for vectors, to distinguish them from ordinary (real or complex) numbers. The notation is

$$\overline{c} = \{c_0, c_1, c_2, \ldots\},$$

where the convolution symbol is on the left and the sequence on the right. The associated Taylor series is

$$c(x) = c_0 + c_1 x + c_2 x^2 + \cdots$$

The name c is the same for the function and the coefficients so that, to every operation with several convolution numbers, we have the associated operation on functions with the same names, which is necessary for consistency in the way the transformation between the two are considered. Actually, we compromise with our notation because we use the same symbol $\overline{c}$ for a vector, and then we use the customary convolution symbol for the product, i.e., $\overline{a} * \overline{b}$. This is apparently not in line with the terminology *number* we have chosen, but we are going to use the same sequence of coefficients as an algebraic vector with the symbol $\overline{c}$ to write the Taylor series as scalar (dot) vector product, which proves to be convenient in the formulation of the theory of convolution numbers, with the $\cdot$ symbol for the scalar product. We are going to use the term *convolution product* as default for *convolution of the vector* $\overline{c}$ or for *product of the convolution number* $\overline{c}$. Otherwise, when we use the term *product* it must be clear from the context whether we mean product with the convolution number or the scalar (dot) product with the vector. Compared to the analog terminology of complex numbers, a product of complex numbers is defined uniquely because complex numbers are not considered as vectors at the same time (except by error). Using vector symbols allows us to use familiar matrix algebra which puts emphasis on the concept of algebra of single quantities, rather than the summation sign $\sum$ which emphasizes internal arithmetic.

In all of the following matrix algebra, we use a consistent matrix tensor notation as described in [25].

We can also write $\underline{c}$ for the convolution number. Particularly, however, we consider $\overline{c}$ a column vector and $\underline{c}$ a row vector. Whenever an abstract vector space is defined, we consider $\overline{c}$ as the contravariant vector and $\underline{c}$ as the covariant vector in some nonorthonormal base. We call $\overline{c}$ and $\underline{c}$ the contravariant and the covariant forms of the convolution number. All our definitions, algebra and analysis that follows could be written in either form. In this treatise, we use the contravariant form, considered as a column vector, simply because it joins best with customary matrix-vector practice. This may be unfortunate because the customary series formula $c(x) = \sum c_i x^i$ is just the other way around, but we see just as many disadvantages in writing the customary matrix-vector algebra the other way round. We use subscripts for the indices of the elements of convolution numbers, which is easier to read than superscripts which is the customary notation for elements of a contravariant vector. This is allowable as long as we don't use covariant vectors in nonorthonormal base in the same context.

2. DEFINITION OF THE CONVOLUTION NUMBER

Let a function $a(x)$ be described within a region of convergence by a Taylor series[3] by

$$a(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \cdots \tag{1}$$

then the sequence of coefficients is a convolution number denoted by the column vector

$$\overline{a} = \left\{ a_0, a_1, a_2, a_3, \ldots \right\}. \tag{2}$$

The sequence is written in a row when space is to be saved, the column character is implied by the vector symbol on the left. Indices of the elements are counted from zero, according to the degree of polynomial. Generally, the function may be a complex valued function $c(z)$, described by a Taylor series within a circle of convergence in the complex z-plane

$$c(z) = c_0 + c_1 z + c_2 z^2 + c_3 z^3 \cdots \tag{3}$$

then the sequence of complex coefficients $\{c_0, c_1, c_2, c_3, \ldots\}$ is a convolution number denoted by the column vector symbol $\overline{c}$. In any case, the radius of convergence in the complex plane must be considered, whether it is known or not. Equation (1) is the connection between a continuous variable and a countable infinite sequence of numbers by which ordinary algebra of continuous functions is transformed to algebra of discrete numbers. The transformation (1) is written, in short, in matrix tensor notation, with the contravariant form of the convolution number, as

$$a(x) = \underline{X} \cdot \overline{a}, \tag{4}$$

where

$$\underline{X} = \left[1, x, x^2, x^3, \ldots \right]. \tag{5}$$

The notation used in equation (4) is a default notation based on the interpretation that the countable infinite number of polynomials $1, x, x^2, x^3, \ldots$, are considered base vectors in a function space, which form a base n in which the function $a(x)$, also a vector in function space, is measured by the vector array $\overline{a}$ of components according to the fully notated equation, using a matrix tensor notation from [25],

$$\vec{a} = \left[\vec{1}, \vec{x}, \vec{x^2}, \vec{x^3}, \ldots \right] \cdot \overline{a} \tag{6}$$

$$= \underline{\vec{X}}_n \cdot \overline{a}^n. \tag{7}$$

Particularly, we may consider the sampled function values $a(x)$ at some infinitely many points on the x-axis within the region of convergence as a vector $\overline{a}^x$ in a reference base called x, so that the transformation equation (6) becomes

$$\overline{a}^x = \left[\overline{1}^x, \overline{x}^x, \overline{x^2}^x, \overline{x^3}^x, \ldots \right] \cdot \overline{a}^n \tag{8}$$

$$= \underline{\overline{X}}^x_n \cdot \overline{a}^n. \tag{9}$$

The corresponding equation for the complex z-plane is

$$\overline{a}^z = \left[\overline{1}^z, \overline{z}^z, \overline{z^2}^z, \overline{z^3}^z, \ldots \right] \cdot \overline{a}^n \tag{10}$$

$$= \underline{\overline{Z}}^z_n \cdot \overline{a}^n, \tag{11}$$

[3]We only use expansions around the origin of the continuous variable, therefore strictly speaking, we are using Maclaurin's series. The theory, however, is not restricted to Maclaurin's series.

where $\overline{a}^{\,z}$ are the function values on some closed line C around the origin, inside the circle of convergence in the z-plane. The purpose of convolution number algebra is to do algebra of Taylor series of continuous functions, therefore, we will never consider a convolution number being the transformation of only a finite number of function values.

Numerical computations with the convolution number imply a finite number of values, which is the truncated sequence of Taylor coefficients. We call the finite sequence as used for practical computations also convolution number, of length n if it is truncated at the n^{th} coefficient, similar to the representation of a real number by a finite sequence of digits in a computer.

The inverse transformation can be written as inverse of equation (11),

$$\overline{a}^{\,n} = \underline{\overline{Z}}_z^{\,n} \cdot \overline{a}^{\,z}, \tag{12}$$

$$\text{where} \qquad \underline{\overline{Z}}_z^{\,n} \equiv \underline{\overline{Z^{-1}}}_n^{\,z}, \quad \text{the inverse of } \underline{\overline{Z}}_n^{\,z}.$$

The dot product here is an integration over the continous functions, *viz.* the Cauchy integral formula

$$\underline{\overline{Z}}_z^{\,n} \cdot \overline{a}^{\,z} = \left\{ \frac{1}{2\pi\hat{\imath}} \int_C \frac{a(z)}{z^{n+1}}, \quad i = 0, \ldots, n \right\}.$$

The inverse transformation integral will never be used in convolution numbers because the very purpose of convolution numbers is to replace analytic forms by numerical computation of coefficients. In the equation (12) the function must be given in terms of a Taylor series first before it can be integrated, i.e., the solution of the integral must be given before the integration can be performed! We are going to use the Taylor transform only formally, merely to relace the phrase: "the coefficients of the function $a(x)$ are ..." in the form of the inverse of equation (4), e.g., in

$$a(x) = \frac{1}{1-x}, \quad \overline{X} \cdot a(x) = \{1, 1, 1, \ldots\}.$$

It is customary to call the result of a transformation operation of equation (4) a transform, whether it is between continuous functions or not, and give such a transform a particular proper name according to the kernel. In this sense, we call $\overline{a}$ in Cauchy's formula of equation (12) the *Taylor transform* of $a(x)$, and $a(x)$ of equation (4) the *inverse Taylor transform* of $\overline{a}$. Note that the whole of the left-hand side of equation (4) is the Taylor series, but $\overline{a}$ is the Taylor transform. In terms of transform language, $a(x)$ and $\overline{a}$ are a transform pair. We also adapt some terminology from Fourier analysis and speak of $a(x)$ in the number *domain* and $\overline{a}$ in the convolution number *domain*, or convolution domain for short, where equation (4) is the transformation from one domain to the other. The complete notation to distinguish between the functional forms would be $a_x(x)$ and $a_n(n)$, [25], but our emphasis is that we will never use the functional form in the convolution number domain and, therefore, we may use $a(x)$ as the default notation for $a_x(x)$ without confusion.

This whole treatise is based on the important difference between the Taylor expansion, or polynomial, $\underline{X} \cdot \overline{a}$ and the convolution number $\overline{a}$. By separating the coefficients from the functions x^n, we change a previously symbolic algebra of polynomials into a numeric algebra of convolution numbers.

The relation of the Taylor transform (12) and other transforms, particularly the z-transform [26], is briefly discussed in Appendix 1.

The Taylor series can be interpreted to be the half-infinite part of a Laurent series. Other half-infinite parts, with a finite number of negative powers of x, of the Laurent series could be included in the convolution number, but this would seriously distract from the simplicity of the convolution number and we would rather handle such cases separately by appropriate transformation to or separation from the Taylor series.

3. ARITHMETIC OF THE CONVOLUTION NUMBER

Arithmetic of the convolution number is defined by the same arithmetic operations on the sum of the terms of polynomials, executed term for term and then collecting coefficients of equal powers as defined by the Taylor expansion of equation (4). The basic arithmetical routines are all well-documented in the literature, [7,8,12,13], or can be easily derived from them, but we revise them here with the numerical aspects in mind; which also means that we don't apply them to formal power series but to real and complex Taylor series inside their circle of convergence.

In this treatise, we take all convolution numbers to the same predetermined length n—similar to digital computation with fixed number of digits measured from the decimal point (fixed point numbers). This means we are essentially doing arithmetic with coefficients of polynomials modulo x^n, [27].

3.1. Addition and Subtraction

$$\text{If } a(x) \pm b(x) = \underline{X} \cdot \overline{a} \pm \underline{X} \cdot \overline{b} = c(x) = \underline{X} \cdot \overline{c}, \quad \text{then}$$

$$\overline{a} \pm \overline{b} = \overline{c}, \tag{13}$$

$$\text{where} \qquad \overline{c} = \{a_0 \pm b_0,\ a_1 \pm b_1,\ a_2 \pm b_2, \ldots\},$$

$$\text{or in indicial notation} \qquad c_i = a_i \pm b_i, \qquad \text{for } i = 0, 1, \ldots, n. \tag{14}$$

3.2. Multiplication

$$\text{If } a(x) \cdot b(x) = \left(\underline{X} \cdot \overline{a}\right) \cdot \left(\underline{X} \cdot \overline{b}\right) = c(x) = \underline{X} \cdot \overline{c}, \quad \text{then}$$

$$\overline{a} * \overline{b} = \overline{c}, \tag{15}$$

where the multiplication is indicated by $*$ which is the customary symbol for convolution of sequences, particularly also in the algebra of sequences [10]. In order to see clearly how to collect all the terms that occur in the product, we show the terms of $\overline{a} * \overline{b}$ in a dyadic array in Figure 1. The convolution number is stored in an array length $n + 2$ (counted from zero), where the elements 0 to n are stored in the first n positions. Figure 1 shows the convolution number $\overline{a}$ as a horizontal and the convolution number $\overline{b}$ as a vertical array. All product terms occur where the vertical and horizontal lines from the elements cross. The diagonal lines connect coefficients of equal powers, and summation along these lines produce the coefficients of $\overline{c}$. Truncating all convolution numbers at n, only the upper left triangle of the dyadic array contributes to the convolution number $\overline{c}$. By following this rule the familiar form of the convolution product is obtained

$$c_i = \sum_{j=0}^{i} a_j\, b_{i-j} = \sum_{k=0}^{i} a_{i-k}\, b_k, \qquad \text{for } i = 0, 1, 2, \ldots, n, \tag{16}$$

depending on whether the summation along the diagonal runs from lower left to upper right or from upper right to lower left. In the determination of the indices of $\overline{a}$ and $\overline{b}$ that make up an element i of $\overline{c}$, we can use the indices j of $\overline{a}$ and k of $\overline{b}$ like coordinates of position of the element i, measured from the origin at the upper left of the dyadic array, such that $j + k = i$, e.g., $(j = 3) + (k = 5) = (i = 8)$ in Figure 1, taking advantage of the fact that the indices start at 0. Although we don't discuss efficiency of computer algorithms of arithmetic operations in this treatise, the user must be aware of it and we will delete at least the obvious zero operations.

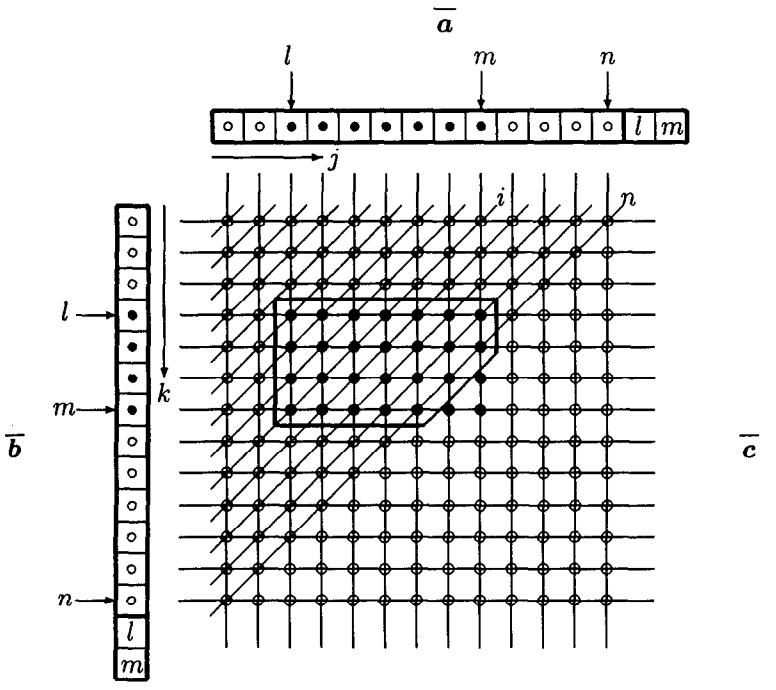


Figure 1. Convolution number multiplication $\overline{a} * \overline{b} = \overline{c}$.

To this end, the last two positions contain the pointers

$$l = \text{position of the leading nonzero term,} \tag{17}$$

$$m = \text{position of the last nonzero term.} \tag{18}$$

In the two convolution numbers $\overline{a}$ and $\overline{b}$, all leading and trailing zero elements are indicated by a $\circ$ and nonzero elements by a $\bullet$, and similarly the resulting zero and nonzero elements in the dyadic array, from which it can be seen that the nonzero elements of $\overline{c}$ are only contained in a smaller nonzero rectangle of the dyadic array, which may also be truncated by the length n. By adding the coordinates of the upper left of the nonzero block in the dyadic array, the position of the leading (nonzero) and last (nonzero) element of $\overline{c}$ is

$$l_c = \min(l_a + l_b, n+1) (5 = 2 + 3 \text{ in Figure 1})$$
$$m_c = \min(m_a + m_b, n) (12 = \min(8 + 6, 12) \text{ in Figure 1}).$$

If $l_a + l_b > n$, then the result is truncated as apparent zero which must be considered in the programming. The lower and upper bounds of the running index j or k are found by the same method from the figure, depending on which side of the nonzero rectangle the diagonal line for the summation enters or leaves for each i,

$$j_1 = \max(l_a, i - m_b), \qquad k_1 = \max(l_b, i - m_a),$$
$$j_2 = \min(i - l_b, m_a), \qquad k_2 = \min(i - l_a, m_b).$$

Therefore, the operations in the familiar multiplication formula of equation (16) are reduced to

$$c_i = \sum_{j=j_1}^{j_2} a_j \, b_{i-j} = \sum_{k=k_1}^{k_2} a_{i-k} \, b_k, \qquad \text{for } i = l_c, \ldots, m_c. \tag{19}$$

By this formula, no computer time is wasted if we allow space for long numbers but have many multiplications with very short numbers, e.g., with only one nonzero position which often occurs in some convolution numbers of a problem. As soon as the new number is computed it has to be scanned from the upper end for a possible new upper bound m_c and then the indices l_c, m_c are inserted in the computer storage of $\overline{c}$. We note the well-known fact that even if $\overline{a}$ and $\overline{b}$ are infinitely long convolution numbers corresponding to the Taylor expansion of a nonpolynomial function, all the computed elements c_i consist of a finite number number of terms only, and as such are exact, or machine exact inasmuch the numbers in $\overline{a}$ and $\overline{b}$ are machine exact. For complex numbers, the formula may be the same, but if there are many occurrences of purely real or purely imaginary convolution number parts, then separate computation of parts may be done, each with their own limits. However, for further discussion, we always assume that l and m are lower and upper bounds of real or complex convolution numbers, not of parts.

3.3. Division

The algorithm for division of convolution numbers follows simply by solution of the multiplication problem. Again, we compute resulting coefficients numerically as soon as possible, in contrast to classic analytic division [28]. Referring to Figure 1, let $\overline{a}$ be the unknown in the equation $\overline{a} * \overline{b} = \overline{c}$. We use the convolution division sign /, there can hardly be any confusion with vectors, following [10], and write the corresponding division problem:

$$\text{if } c(x)/b(x) = (\underline{X} \cdot \overline{c})/(\underline{X} \cdot \overline{b}) = a(x) = \underline{X} \cdot \overline{a} \quad \text{then}$$
$$\overline{c} \,/\, \overline{b} = \overline{a}. \tag{20}$$

Referring to Figure 1, and assuming for the moment that all positions in the dyadic array have nonzero elements, it can be seen by following a diagonal of the terms of the element c_i that the highest element of $\overline{a}$ which contributes to c_i is a_i. Therefore, if all elements of $\overline{a}$ below i are known, we can compute a_i by the formula

$$a_0 = c_0/b_0,$$
$$a_i = \left(c_i - \sum_{j=0}^{i-1} a_j b_{i-j}\right)/b_0 = \left(c_i - \sum_{k=1}^{i} a_{i-k} b_k\right)/b_0, \qquad \text{for } i = 1, 2, \ldots, n. \tag{21}$$

The customary series division requires that the leading coefficient $b_0 \neq 0$, [7,12,27,29], but this is unnecessarily restrictive, we wouldn't even be able to expand the Taylor series of such a well-known function as $\sin(x)/x$. The beauty of convolution number division is now that we can just as easily divide by numbers with leading zeros—as long as they are compatible.

Polynomials with nonzero leading elements are called units and with zero leading element nonunit [8]. These terms don't suit us because applied to the vector interpretation of a convolution number, they convey a different meaning. We call the convolution number with nonzero leading element *regular* and with any number of leading elements *singular*, the singular feature being that it cannot be inverted.

The calculation of the leading nonzero position from Figure 1 becomes now

$$l_a = l_c - l_b.$$

Therefore, to obtain a convolution number after division, $l_a \geq 0$, the compatibility condition for

division is[4]

$$l_c - l_b \geq 0. \tag{22}$$

Referring again to Figure 1, the limits of $\overline{a}$ in the division of equation (21) can be set and the division routine limited to operations of nonzero numbers

$$\begin{aligned} &l_a = l_c - l_b \text{ must be} \geq 0 \\ &j_1 = \max(l_a, i - m_b), \qquad k_1 = l_b + 1 \\ &j_2 = i - 1, \qquad\qquad k_2 = \min(i + l_b - l_a, m_b) \\ a_i = &\left(c_{i+l_b} - \sum_{j=j_1}^{i-1} a_j\, b_{i+l_b-j} \right) / b_{l_b} = \left(c_{i+l_b} - \sum_{k=l_b+1}^{k_2} a_{i+l_b-k}\, b_k \right) / b_{l_b} \end{aligned} \tag{23}$$

$$\text{for } i = l_a, \ldots, (n - l_b).$$

The highest possible nonzero position of $\overline{a}$ cannot be determined beforehand, in fact, it will be infinity (n) generally, but because of the leading zeros in $\overline{b}$, it cannot be determined to more than length $n - l_b$. We could determine some numbers in that range, but they would not be unique because they depend on numbers that are truncated in $\overline{b}$ or $\overline{c}$. The actual computer stored number $\overline{a}$ has, therefore, trailing zeros which are not true zeros; this is the price we pay for being able to do division with leading zero numbers and truncating at constant length n. If this is a concern in applications then convolution numbers could be programmed with a floating leading position—similar to floating point numbers in digital computers.

Nevertheless, there may be more trailing zeros and the resulting number must be scanned from above after division to determine the actual higher limit m_a.

Several features are interesting in the division formula. The leading element position is not only useful for avoiding operations with zeros but also necessary to determine which divisions are possible, which is equivalent to determining the leading element by L'Hospital's rule. Other than in the previous arithmetic routines, the elements of of the unknown number $\overline{a}$ are used in the division that determines $\overline{a}$ and therefore only one sequence to determine a_i can be followed. Yet the previous property still exists, that each element of the result consists only of a finite number of terms of the contributing numbers, and is, therefore, exactly independent of how the convolution number is truncated. Specifically, we find that in a sequence of multiplication and division we obtain the exact original numbers back, even though much information has apparently been lost by overflow, which can be regarded as a test that the algebra of multiplication and division is consistent.

EXAMPLE. Division of the Taylor series $x/\left(x + x^2 + x^3 + x^4 + \cdots\right)$ corresponds to convolution number division, using length 10,

$$\{0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,|\;1\,1\}/\{0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,|\;1\,1\} = \{1\,-1\,0\,0\,0\,0\,0\,0\,0\,0\,|\;0\,1\}.$$

Division the other way around, starting with the division of the highest order element in the numerator by the highest order element of the denominator, called long division or synthetic division in [27], is not allowed for two reasons. First, it leads to a remainder term, and if continued, to a Laurent series. Secondly, the highest order terms are the worst numerical approximations of the theoretical Taylor series.

To allow for compatible division in a program, all multiplication should be done before division, e.g., $(\cos(x)\sin(x))/x$ is possible, but $(\cos(x)/x)\sin(x)$ is not. Also, the denominator may have to be taken out as factor, e.g., $(x - \sin(x))/x^3$ is possible, but $x/x^3 - \sin(x)/x^3$ is not.

[4]Division by a convolution number with a larger number of leading zeros would be possible if we extended the Taylor series to a finite number of Laurent series terms, but this is a complication that is rather avoided by handling such cases separately, as is done in Example 1. This corresponds to avoiding the numbers $1/0, 1/0^2, \ldots,$ in real number theory.

3.4. Square Root

We use the same routine as for division by solving the equation $\overline{a} * \overline{a} = \overline{c}$ for $\overline{a}$ with slightly different limits and initial element. The condition for compatability w.r.t. leading zeros is now that l_c must be an integer divisible by two. Adding the nonzero terms along the diagonal in Figure 1, which are known from the previous steps, we arrive at the formula

$$\begin{gathered} l_a = l_c/2 \text{ must be an integer,} \\ a_{l_a} = \sqrt{c_{l_c}}, \quad a_i = \left(c_{l_a+i} - \sum_{j=l_a+1}^{i-1} a_j \, a_{i+l_a-j} \right) / (2a_{l_a}), \\ \text{for } i = l_a + 1, \ldots, (n - l_a). \end{gathered} \tag{24}$$

The actual programming should take advantage of the symmetry so that only almost half of the multiplications have to be done. Another new feature occurs in this operation. The leading element equation has exactly two possible solutions, generally considering the complex case, but all subsequent elements have only one solution depending on the initial, therefore, the convolution number square root has two and only two distinct solutions. The case of a real convolution number, $\overline{a} = \pm\sqrt{\overline{c}}$ is reflected not in applying the opposite sign to all elements to obtain the second solution, but to apply it only to the leading element. The formula equation (24) shows how the sign of the leading element is transmitted to all other elements. In the general complex case, the elements of the two roots do not have the simple $\pm$ relation any more.

The square root in our point of view is not just a special case of an integer root and, therefore, has its own place in the arithmetic. Other integer roots can be done with the routine for the general exponential with any real or complex number which is introduced later.

3.5. Exponentiation

In this section, we indicate possible algorithms using the previous arithmetic merely to define powers of a convolution number, the actual methods will be given much later. For the power of a convolution number $\overline{a}$ with a scalar exponent k we use the symbol $\overline{a}^{*k}$. The integral power is defined by

$$\overline{a}^{*k} = \overline{a} * \overline{a} * \overline{a}, \ldots, k \text{ factors.} \tag{25}$$

For the power with a general scalar s, we assume in the meantime that it can be broken up into factors with integral powers and square roots, and for negative exponents the corresponding division.

3.6. Special Numbers

To have a complete system of convolution numbers, we must allow for a convolution zero $\overline{0}$, and a convolution unit $\overline{\delta}$; these are denoted by 0 and δ in [10], respectively. In all, we use the special symbols

$$\begin{aligned} \overline{\delta}_0 &= \{1, 0, 0, \ldots\} \\ \overline{\delta}_1 &= \{0, 1, 0, \ldots\} \\ \overline{\delta}_i &= \{0, \ldots, 1\,(i^{\text{th}}\text{position}), 0, \ldots\} \\ \overline{0} &= \{0, 0, \ldots\} \\ \overline{1} &= \{1, 1, 1, \ldots\} \\ \overline{1}_i &= \{0, \ldots, 1\,(i^{\text{th}}\text{position}), 1, 1, \ldots\}, \end{aligned} \tag{26}$$

where $\overline{\delta}$ and $\overline{1}$ are the defaults for δ_0 and $\overline{1}_0$, respectively. $\overline{\delta}_k$ and $\overline{1}$ are denoted by p , p^k and l in [10], respectively. The $\overline{\delta}_1$ plays a special role because it represents the variable itself in the transformation $\underline{X} \cdot \overline{\delta}_1 = x$. Also, $\overline{\delta}_i$ has the special property that multiplication with a convolution number $\overline{c}$ produces a shift by i positions of the whole sequence $\overline{c}$ towards the end, inserting i leading zeros, which can be seen from the multiplication diagram Figure 1. This property is useful in some applications.

The scalar quantity s can be regarded as a special case of a polynomial and corresponds therefore to the convolution number $\overline{s} = \{ s, 0, \ldots \}$. Obviously, arithmetic operations with a scalar s and a Taylor series $a(x)$ correspond then to the same operations on every element a_i of the corresponding convolution number $\overline{a}$, which on the other hand are equal to the convolution arithmetic operations of the convolution number $\overline{s}$ and $\overline{a}$. To simplify these steps, arithmetical operations between a scalar and a convolution number are defined in the above sense, i.e., eliminating the need to transform the scalar to a convolution number, an idea which is also used by Mikusiński [11]. The convolution number corresponding to the scalar s is then expressed by the now defined product $s\,\overline{\delta}$, as used in [10]. Note that the symbol $\overline{s}$ is not the transform of a scalar number but rather of a function $s(x)$.

3.7. Programming Considerations

The formulas, as given, assume that the initial value of all elements of a resulting convolution number are zero, and the summation is not executed if the upper index is smaller than the lower index, and the whole operation is not executed if the final value for the index i is smaller than the initial value. For this reason, the limit l_c is taken as $n + 1$ in the multiplication routine if the leading zeros of $\overline{a}$ and $\overline{b}$ already cause overflow. It has been shown how important the lower limits l are to include compatible division and square root, and how useful the upper limits m are. They must therefore be reset in the result after each operation, inside the routine, particularly the lower limit l is unpredictable in the formulas for addition and subtraction. A small radius ϵ should be chosen within which a number is zero for this purpose. A complete system of convolution number algebra could be programmed within a PC language similar to the matrix algebra that was available in the earlier BASIC implementations, which was written with the prefix MAT, so that statements would look like

```
CON C = A + B
CON C = A * B
CON C = C/B
CON C = SQR(B)
CON C = A^S.
```

Without such implementation, the arithmetic operations are written as subroutines to be CALLed, which should have proper short names that indicate their operation, and the subroutines and variables may have the term *con* as beginning or end of their names to simulate the mathematical convolution number symbol. By putting the pointers into the number storage, not into the arguments of a calling routine, gives a neat appearance, and it allows programming of a mathematical language. Other numbers can be stored at the end, e.g., to allow a floating length, similar to the E-information of floating point numbers in a digital computer. Programming must also allow for the fact that customary computer programming allows the result to overwrite one of the initial numbers. To make programs well-readable, the initial setting of elements and pointers should also be done in a subroutine so that initializing does not occupy more than a single line in the main program. Similar to the error messages in number programs, error messages should

appear and the program stopped if incompatible conditions in division or square root routine occur. Setting of the lower and upper limits l and m should be done in the subroutines to relieve the programmer from tasks that have to do with the internal structure. Checking and error messages should be included to prevent wrong settings if the programmer accesses the convolution number directly instead of using a setting routine.

3.8. Other Methods

The convolution number product $\overline{a} * \overline{b}$ can be written as a matrix · vector product $\underline{\overline{A}} \cdot \overline{b}$ where the matrix $\underline{\overline{A}}$ consists of all the elements of $\overline{a}$ arranged as coefficients of $\overline{b}$. This associated matrix is semicirculant as described in [8], and its triangular property can be used to explain the division process as the solution of the multiplication process which becomes the backsubstitution part of a Gauss elimination routine. This algebra of the associated matrix is not used in numerical applications because the numbers do not need matrix storage. Yet the semicicular matrix and a more general triangular matrix will appear later in the theory.

Another method that has been suggested is using FFT routines, [16,17], but that method cannot be used in our convolution number arithmetic. The FFT routines need the evaluation of the function on a circle in the complex plane at all n points, which was probably taken at unit radius in [17], therefore, we can never take the advantage of leading and trailing zeros, although admittedly the FFT method may be faster in the average. But the FFT method produces aliased coefficients, which destroys the property of exact Taylor coefficients. The classic convolution number arithmetic as presented above can also be used to determine coefficients analytically while the FFT method does not allow this. So far, these are not serious objections to a numerical method. A serious problem is that we generally don't know the radius of convergence. The FFT may use a radius which is many orders too large or many orders too small, which, theoretically, would still be insignificant if a truncated Taylor expansion is analyzed, but numerically, in both cases, significant coefficients are lost. The convolution number itself does not have to be a convergent sequence at all, as in an example that follows, it is the Taylor series within the circle of convergence that has to converge. The purpose of convolution numbers is to work well inside the circle of convergence. On the other hand if FFT is used to find the coefficients of a function outside the circle of convergence of the Taylor series, then the Laurent series on that circle which now encloses a singularity, is found instead.

4. CONVOLUTION MATRIX

4.1. Definitions

The internal arithmetic of convolution number algebra is defined by addition/subtraction, multiplication and division of elements. Generalizing, the elements can be anything on which these operations are defined. Consider these elements being matrices,

$$\overline{a} = \left\{ \underline{\overline{A}}_0,\ \underline{\overline{A}}_1,\ \underline{\overline{A}}_2, \dots \right\}, \tag{27}$$

$$\overline{b} = \left\{ \underline{\overline{B}}_0,\ \underline{\overline{B}}_1,\ \underline{\overline{B}}_2, \dots \right\}. \tag{28}$$

Such a *matrix convolution number* is shown diagrammatically in Figure 2(a), with p as the running index. The sum $\overline{a} + \overline{b}$ is defined by the matrix addition of the elements. The product is defined

by the convolution product

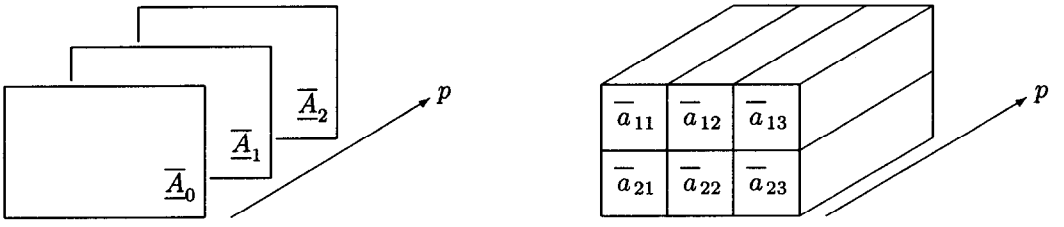
$$\overline{a} * \overline{b} = \begin{bmatrix} \overline{A}_0 \cdot \overline{B}_0 & \\ \overline{A}_1 \cdot \overline{B}_0 + \overline{A}_0 \cdot \overline{B}_1 & \\ \overline{A}_2 \cdot \overline{B}_0 + \overline{A}_1 \cdot \overline{B}_1 + \overline{A}_0 \cdot \overline{B}_2 & \\ \vdots & \ddots \end{bmatrix}, \tag{29}$$

where the internal multiplications are matrix multiplications. Obviously, this convolution algebra is limited by the rules of matrix algebra, i.e., matrices must be conformable, division is only possible if matrices are square, and, particularly, multiplication is not commutative, $\overline{a} * \overline{b} \neq \overline{b} * \overline{a}$.

Alternatively, consider a matrix with elements that are convolution numbers

$$\underline{\overline{A}} = \begin{bmatrix} \overline{a}_{11} & \overline{a}_{12} & \overline{a}_{13} & \dots \\ \overline{a}_{21} & \overline{a}_{22} & \overline{a}_{23} & \dots \\ \vdots & & & \end{bmatrix}, \quad \underline{\overline{B}} = \begin{bmatrix} \overline{b}_{11} & \overline{b}_{12} & \overline{b}_{13} & \dots \\ \overline{b}_{21} & \overline{b}_{22} & \overline{b}_{23} & \dots \\ \vdots & & & \end{bmatrix}.$$

Such a *convolution number matrix* is shown diagrammatically in Figure 2(b). The dimensions of the matrix can be any $m \times n$, including column and row vectors.



(a) Matrix Convolution Number

(b) Convolution Number Matrix

Figure 2.

The internal arithmetic of the matrix multiplication $\underline{\overline{A}} \cdot \underline{\overline{B}}$ consists of products of convolution numbers

$$\underline{\overline{A}} \cdot \underline{\overline{B}} = \sum \overline{a}_{ik} * \overline{b}_{kj}, \quad i = 1, \dots, m, \quad j = 1, \dots, n. \tag{30}$$

Inserting the convolution product formula of Section 3, the finite sums of equation (30) can be rearranged to be equal to the terms of equation (29). The same can be done for the left and right matrix division, as defined in [30], being Gauss elimination, and we arrive at the conclusion that a convolution number matrix is the same as a matrix convolution number. It may be that the applications favour the one or other approach. Both we simply call *convolution matrix*, and we call the matrices in Figure 2(a) *slices* of the convolution matrix in Figure 2(b). In any case, the matrix algebra is the restrictive algebra and we don't attempt to take the convolution number matrix algebra any further than the four arithmetical operations, and we keep in mind that equations (4) and (12) apply to each element of the matrix, which is formalized by Theorems 3 and 4 in Section 11.

We denote the convolution matrix by the symbol $\underline{\overline{\overline{A}}}$, where the outer bars are the matrix symbols and the inner upper bar is the convolution number symbol. Multiplication is written

with the convolution symbol $*$. For left and right division, we use the ordinary division symbols superimposed with the convolution symbol as $\backslash\!\!*$ and $*\!\!/$, respectively. These symbols are actually an overemphasis of the operation because no other is possible, but it allows to write in default notation, e.g., $A * b$, $C \backslash\!\!* B$, or to use the boldface tensor-vector notation $\mathbf{A} * \mathbf{b}$, $\mathbf{C} * \mathbf{B}$, where now the convolution operation indicates the convolution tensor-vector.

For later reference, we also state the convolution matrix left and right division in terms of matrix algebra. If $\underline{\overline{A}} = \underline{\overline{B}} \backslash\!\!* \underline{\overline{C}}$, then

$$\begin{aligned}
\overline{\underline{A}}_0 &= \overline{\underline{B}}_0 \backslash \overline{\underline{C}}_0 \\
\overline{\underline{A}}_1 &= \overline{\underline{B}}_0 \backslash \left[\overline{\underline{C}}_1 - \overline{\underline{B}}_1 \cdot \overline{\underline{A}}_0 \right] \\
\overline{\underline{A}}_2 &= \overline{\underline{B}}_0 \backslash \left[\overline{\underline{C}}_2 - \overline{\underline{B}}_1 \cdot \overline{\underline{A}}_1 - \overline{\underline{B}}_2 \cdot \overline{\underline{A}}_0 \right] \\
&\vdots
\end{aligned} \tag{31}$$

If $\underline{\overline{A}} = \underline{\overline{C}} *\!\!/ \underline{\overline{B}}$, then

$$\begin{aligned}
\overline{\underline{A}}_0 &= \overline{\underline{C}}_0 / \overline{\underline{B}}_0 \\
\overline{\underline{A}}_1 &= \left[\overline{\underline{C}}_1 - \overline{\underline{A}}_0 \cdot \overline{\underline{B}}_1 \right] / \overline{\underline{B}}_0 \\
\overline{\underline{A}}_2 &= \left[\overline{\underline{C}}_2 - \overline{\underline{A}}_1 \cdot \overline{\underline{B}}_1 - \overline{\underline{A}}_0 \cdot \overline{\underline{B}}_2 \right] / \overline{\underline{B}}_0 \\
&\vdots
\end{aligned} \tag{32}$$

Particularly, the inverse $\underline{\overline{A}} = \underline{\overline{B}}^{*-1}$ is explicitly

$$\begin{aligned}
\overline{\underline{A}}_0 = \overline{\underline{B}}_0^{-1} \qquad &= \quad \overline{\underline{B}}_0^{-1} \\
\overline{\underline{A}}_1 = -\overline{\underline{A}}_0 \cdot \overline{\underline{B}}_1 \cdot \overline{\underline{A}}_0 \qquad &= \quad -\overline{\underline{B}}_0^{-1} \cdot \overline{\underline{B}}_1 \cdot \overline{\underline{B}}_0^{-1} \\
\overline{\underline{A}}_2 = \left[-\overline{\underline{A}}_1 \cdot \overline{\underline{B}}_1 - \overline{\underline{A}}_0 \cdot \overline{\underline{B}}_2 \right] \cdot \overline{\underline{A}}_0 \qquad &= \quad \overline{\underline{B}}_0^{-1} \cdot \overline{\underline{B}}_1 \cdot \overline{\underline{B}}_0^{-1} \cdot \overline{\underline{B}}_1 \cdot \overline{\underline{B}}_0^{-1} - \overline{\underline{B}}_0^{-1} \cdot \overline{\underline{B}}_2 \cdot \overline{\underline{B}}_0^{-1},
\end{aligned} \tag{33}$$

and like in ordinary matrix algebra, the left inverse and right inverse are the same. The first expansion shows the numerical implementation while the second is the customary analytical form.

The convolution matrix division can also be written as Gauss elimination with convolution operations replacing the number operations; then it will be found that compatible division may also be possible with singular leading matrix $\overline{\underline{B}}$, in which case the forms of equations (32) and (33) cannot be applied.

In the following, the application to vibration analysis is shown, where the approach starts from the matrix algebra.

4.2. Convolution Matrix in Vibration Analysis

The dynamic equation for a finite degree of freedom (DOF) linear mechanical system is the familiar equation of structural dynamics, [31–33]

$$\overline{\underline{K}} \cdot \overline{u}(t) \; + \; \overline{\underline{C}} \cdot \dot{\overline{u}}(t) \; + \; \overline{\underline{M}} \cdot \ddot{\overline{u}}(t) = \overline{f}(t), \tag{34}$$

where $\overline{u}(t)$, $\overline{f}(t)$ are displacement and applied force vectors, respectively, as function of time, and $\overline{\underline{K}}$, $\overline{\underline{C}}$, $\overline{\underline{M}}$ are the stiffness, damping and mass matrices, respectively, written in the matrix

tensor notation [25]. In vibration analysis, a Fourier transform of equation (34) is made which changes the functions of time to functions of frequency ω

$$\left[\overline{\underline{K}} + \hat{\imath}\omega\overline{\underline{C}} - \omega^2\overline{\underline{M}} \right] \cdot \overline{u}(\omega) = \overline{f}(\omega), \tag{35}$$

e.g., [31], which is the familiar form from which vibration analysis starts.[5] The rhs. of equation (35) is a polynomial matrix [34], of which the general form is usually written in the form $\overline{\underline{A}}_0 + \overline{\underline{A}}_1\lambda + \overline{\underline{A}}_2\lambda^2 \ldots$, which can alternatively be interpreted as a matrix of polynomials, which is the way we interpret equation (35) here. Therefore, we need a collective name for all matrices, for which we choose S derived from (mechanical) *system* matrix, so that equation (35) becomes

$$\overline{\underline{S}}(\omega) \cdot \overline{u}(\omega) = \overline{f}(\omega), \tag{36}$$

where $\overline{\underline{S}}(\omega) = \overline{\underline{K}} + \hat{\imath}\omega\overline{\underline{C}} - \omega^2\overline{\underline{M}}$ is the complex system matrix, also called dynamic stiffness matrix, [35], or impedance matrix [31], denoted by $\mathbf{Z}$. For each polynomial, we write the Taylor transform

$$S_{ij}(\omega) = \underline{W}_p \cdot \overline{S}^p{}_{ij}, \tag{37}$$

where $\underline{W}_p = [1, \omega, \omega^2, \ldots]$ is the base p of polynomials.

We write now the Taylor transform of equation (36) as

$$\overline{\overline{\underline{S}}} * \overline{\overline{u}} = \overline{\overline{f}}, \tag{38}$$

which is understood to be the default notation for the completely notated equation

$$\underline{\overline{\overline{S}}}{}^{m}_{m} * \overline{\overline{u}}{}^{m}_{p} = \overline{\overline{f}}{}^{m}_{p}, \tag{39}$$

where m is the abstract base for the system vectors [36]. $\overline{\overline{\underline{S}}}$ is the complex valued system convolution matrix of which the matrices $\overline{\underline{K}}$, $\hat{\imath}\overline{\underline{C}}$, and $-\overline{\underline{M}}$ are slices. Note that the difference between $\overline{\underline{S}}(\omega)$, written as polynomial, and $\overline{\overline{\underline{S}}}$ is that the first is a single analytical quantity, including the variable ω, while the second is a sequence, purely numerical, suitable for digital computer programming. Convolution matrix routines would generally not be available to the structural dynamics programmer who works with the matrix slices $\overline{\underline{K}}$, $\overline{\underline{C}}$, $\overline{\underline{M}}$ separately and is limited to expansions only up to ω^2.

From equation (36), we may write the vector derivative

$$\frac{d\overline{f}}{d\overline{u}}(\omega) = \overline{\underline{S}}(\omega), \tag{40}$$

and from equation (38) the convolution vector derivative which is then the Taylor transform of equation (40)

$$d\overline{\overline{f}} \not{*} d\overline{\overline{u}} = \overline{\overline{\underline{S}}}. \tag{41}$$

In many applications in structural dynamics, the system matrix is partitioned when some of the forces are zero, which we write in convolution matrix notation

$$\left[\begin{array}{c|c} \overline{\overline{\underline{S}}}_{11} & \overline{\overline{\underline{S}}}_{12} \\ \hline \overline{\overline{\underline{S}}}_{21} & \overline{\overline{\underline{S}}}_{22} \end{array}\right] \cdot \left[\begin{array}{c} \overline{\overline{u}}_1 \\ \hline \overline{\overline{u}}_2 \end{array}\right] = \left[\begin{array}{c} \overline{\overline{f}}_1 \\ \hline \overline{\overline{0}} \end{array}\right]. \tag{42}$$

[5] The symbol $\hat{\imath}$ for the imaginary unit was introduced in [25].

By eliminating $\overline{u}_2$, the relation between $\overline{f}_1$ and $\overline{u}_1$ can be found

$$d\overline{\overline{f}}_1 \not\ast d\overline{\overline{u}}_1 = \overline{\overline{S11}} = \overline{\overline{S}}_{11} - \overline{\overline{S}}_{12} \cdot \overline{\overline{S}}_{22}^{*-1} \cdot \overline{\overline{S}}_{21}. \tag{43}$$

Here, the numerals 11 are part of the name of the condensed matrix so as not to confuse it with partition 11. The result can be obtained by condensation

$$\overline{\overline{S11}} = \overline{\overline{L}}^{\mathsf{T}} * \overline{\overline{S}} * \overline{\overline{L}}, \tag{44}$$

where the condensation matrix is given by

$$\overline{\overline{L}} = \begin{bmatrix} I \\ \hline -\overline{\overline{S}}_{22}^{*-1} * \overline{\overline{S}}_{21} \end{bmatrix}. \tag{45}$$

We have replaced the customary name C of the condensation matrix with the name L to avoid conflict of notation with the damping matrix C.

The original Guyan reduction, [37], was made with the stiffness matrix elements in the condensation matrix only, called static condensation, [33], but it happens to give the same result up to the first slice matrix (counted from zero). Since damping was neglected, $\overline{C} = \overline{0}$, the result was correct to ω^2 which is all that is required in customary structural dynamics. The higher powers can only be obtained by the correct condensation matrix of equation (45), as pointed out in [33], who gave the explicit terms of the condensed matrix up to ω^4 (λ^2 in that notation). However, if damping is considered the slices of the proper condensed matrix contain the damping matrix $\overline{C}$, and are

$$\overline{K11} \equiv \overline{S11}_{p0} = \overline{K}_{11} - \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21} \quad \text{(as before)},$$

$$\overline{C11} \equiv -\hat{\imath}\overline{S11}_{p1} = \overline{C}_{11} - \overline{C}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21} - \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{21} + \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21},$$

$$\begin{aligned} \overline{M11} \equiv -\overline{S11}_{p2} = {} & \overline{M}_{11} - \overline{M}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21} - \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{M}_{21} + \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{M}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21} \\ & - \overline{C}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{21} + \overline{C}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21} \\ & + \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{21} - \overline{K}_{12} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{C}_{22} \cdot \overline{K}_{22}^{-1} \cdot \overline{K}_{21}. \end{aligned}$$

Here, the double numeric subscripts refer to the system matrix partitions, and the subscripts preceded by the letter p refer to convolution matrix slices. The additional terms due to the damping matrix are consistent with the assumption of damping, and it is interesting to see how damping couples with the stiffness matrix. Static condensation would miss all the additional damping terms in the condensed mass matrix. The fact that in most applications of structural dynamics the damping is known only very approximately is a separate issue.

It is obvious that the explicit computation of the formulas above is very inefficient compared to convolution routines implemented directly in equation (43), especially if expansion to higher powers is desired. This is not done in practice though [33]. The truncated condensed system matrix must be regarded as an approximation rather than an analytical expansion. A singularity appears due to division corresponding to the lowest eigenfrequency of the partitioned system $\overline{\overline{S}}_{22}$. It is surmised that the lower eigenfrequencies of the condensed matrix are much lower, this is even more so if the condensed matrix is only one element in an assembly of finite elements, where even the highest computed frequency of the assembly is below the lowest frequency of the elements. Customary finite element practice deals only with at most the three matrices $\overline{K}, \overline{C}, \overline{M}$ and has

no provision for a form like the theoretical condensed matrix, which obviously doesn't allow a distinction between stiffness and mass matrix any more.

As another example, we consider a sructural element which is known analytically and then approximated by a finite element. A uniform rod of length l, crossection A, density ρ, modulus of elasticity E has the integral properties mass $m = \rho Al$, stiffness $k = EA/l$. The system matrix of displacements and forces at the ends is obtained analytically [35], without damping,

$$\underline{\overline{S}}(\omega) = \frac{EA}{l} \begin{bmatrix} \frac{\gamma \cos\gamma}{\sin\gamma} & \frac{-\gamma}{\sin\gamma} \\ \frac{-\gamma}{\sin\gamma} & \frac{\gamma\cos\gamma}{\sin\gamma} \end{bmatrix}, \tag{46}$$

where $\gamma = \omega\sqrt{\frac{\rho Al}{EA/l}} = \omega\sqrt{\frac{m}{k}}$. This matrix can actually be interpreted as a condensed system matrix from a system of infinitely many DOF's, originally desribed by a differential equation. Expansion of each element into a Taylor series of γ is valid below the first singularity due to division by $\sin\gamma$ at $\gamma = \pi$, which corresponds to the singularity of the matrix $\underline{\overline{S}}_{22}^{-1}$ in the condensed matrix. The Taylor series' in this case are simple, and the expansion results in

$$\underline{\overline{S}}(\omega) = \frac{EA}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} - \omega^2\frac{\rho Al}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} - \omega^4\frac{(\rho Al)^2 l}{45EA}\begin{bmatrix} 1 & \frac{7}{8} \\ \frac{7}{8} & 1 \end{bmatrix}\ldots \tag{47}$$

$$= \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} - \omega^2\begin{bmatrix} \frac{1}{3}m & \frac{1}{6}m \\ \frac{1}{6}m & \frac{1}{3}m \end{bmatrix} - \omega^4\begin{bmatrix} \frac{1}{45}\frac{m^2}{k} & \frac{7}{360}\frac{m^2}{k} \\ \frac{7}{360}\frac{m^2}{k} & \frac{1}{45}\frac{m^2}{k} \end{bmatrix}\ldots \tag{48}$$

$$\equiv \underline{W}\cdot\underline{\overline{S}}.$$

These matrices in equation (47) are given in [32], but in the strange form of $[k_0] - \omega^2[m_0] - \omega^4[k_4 - m_2]$ which results from the approach of defining stiffness and mass matrices rather irrationally for higher powers of ω^2. Also in [32], the corresponding lowest eigenvalue $\frac{\omega^2\rho L^2}{E}$ of $\underline{\overline{S}}(\omega)$ in equation (48) is given as $\frac{1}{2}(-15 + 9\sqrt{5}) = 1.6007$, while the next eigenvalue of $\frac{1}{2}(-15 - 9\sqrt{5}) = \hat{\imath}4.1907$ is simply "rejected" by practical engineering sense. The mathematical reason is now obvious: the first eigenvalue is well within the radius of convergence of $\pi = 3.141593\ldots$ of the Taylor expansion while the second is outside and, therefore, not a valid function value.

After this simple example, we consider the slightly more complicated case. To develop the system matrix for a uniform beam in a plane the numerical path with convolution matrix algebra will be chosen as soon as possible. Let the beam have a length l, bending stiffness EI, crossection area A and density ρ. The Fourier transform of the differential equation of vertical deflection motion is

$$\frac{d^4v}{dx^4} - \omega^2\frac{\rho A}{EI}v = 0,$$

where v is the complex amplitude of the harmonic deflection normal to the beam and x the length variable. The beam is unloaded but the force $f_1(\omega)$ and torque $q_1(\omega)$ act at the left end, and the force $f_2(\omega)$ and torque $q_2(\omega)$ at the right end. Let $\frac{\omega^2\rho A}{EI} = \beta^4$, then the general solution is

$$v = c_1\ \cosh\beta x + c_2\ \sinh\beta x + c_3\ \cos\beta x + c_4\ \sin\beta x.$$

Using a new variable $\gamma = \beta l$, the boundary conditions are expressed in matrix algebra, for the forces

$$\overline{f}(\omega) \equiv \begin{bmatrix} f_1 \\ q_1 \\ f_2 \\ q_2 \end{bmatrix} = \begin{bmatrix} EIv'''(x=0) \\ -EIv''(x=0) \\ -EIv'''(x=l) \\ EIv''(x=l) \end{bmatrix}$$

$$= \frac{EI}{l^3} \begin{bmatrix} 1 & & & \\ & l & & \\ & & 1 & \\ & & & l \end{bmatrix} \cdot \begin{bmatrix} 0 & \gamma^3 & 0 & -\gamma^3 \\ -\gamma^2 & 0 & \gamma^2 & 0 \\ -\gamma^3 \sinh\gamma & -\gamma^3 \cosh\gamma & -\gamma^3 \sin\gamma & \gamma^3 \cos\gamma \\ \gamma^2 \cosh\gamma & \gamma^2 \sinh\gamma & -\gamma^2 \cos\gamma & -\gamma^2 \sin\gamma \end{bmatrix}$$

$$\equiv \underline{\overline{F}}(\omega) \cdot \overline{c}(\omega) \equiv \frac{EI}{l^3} \underline{\overline{d}} \cdot \underline{\overline{Fn}}(\omega) \cdot \overline{c}(\omega),$$

and for the displacements

$$\overline{u}(\omega) \equiv \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} v(x=0) \\ v'(x=0) \\ v(x=l) \\ v'(x=l) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & & \\ & 1/l & & \\ & & 1 & \\ & & & 1/l \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & \gamma & 0 & \gamma \\ \cosh\gamma & \sinh\gamma & \cos\gamma & \sin\gamma \\ \gamma \sinh\gamma & \gamma \cosh\gamma & -\gamma^2 \sin\gamma & \gamma^2 \cos\gamma \end{bmatrix}$$

$$\equiv \underline{\overline{U}}(\omega) \cdot \overline{c}(\omega) \equiv \underline{\overline{d}}^{-1} \cdot \underline{\overline{Un}}(\omega) \cdot \overline{c}(\omega).$$

The diagonal matrix $\underline{\overline{d}}$ is introduced to separate the symbolic part of variables from the numerical, or dimensionless beam matrices indicated by the additional letter n in the matrix name.

The system matrix is defined by

$$\overline{f}(\omega) = \underline{\overline{S}}(\omega) \cdot \overline{u}(\omega),$$

therefore, by eliminating the vector of contants $\overline{c}$, we find

$$\underline{\overline{S}}(\omega) = \underline{\overline{F}}(\omega)/\underline{\overline{U}}(\omega)$$

$$= \frac{EI}{l^3} \underline{\overline{d}} \cdot \underline{\overline{Fn}}/\underline{\overline{Un}} \cdot \underline{\overline{d}}$$

$$\equiv \frac{EI}{l^3} \underline{\overline{d}} \cdot \underline{\overline{Sn}} \cdot \underline{\overline{d}}.$$

The purpose is to determine the numerical matrices in the Taylor expansion

$$\underline{\overline{Sn}} = \underline{\overline{Sn}}_0 + \gamma \underline{\overline{Sn}}_1 + \gamma^2 \underline{\overline{Sn}}_2 \cdots = \underline{G} \cdot \underline{\underline{\overline{Sn}}}.$$

At this stage, we introduce convolution numbers defined by

$$\cosh\gamma = \underline{G} \cdot \overline{hc}, \qquad \gamma = \underline{G} \cdot \overline{\delta}_1,$$

$$\sinh\gamma = \underline{G} \cdot \overline{hs}, \qquad \gamma^2 = \underline{G} \cdot \overline{\delta}_2,$$

$$\cos\gamma = \underline{G} \cdot \overline{cs}, \qquad \gamma^3 = \underline{G} \cdot \overline{\delta}_3,$$

$$\sin\gamma = \underline{G} \cdot \overline{sn}.$$

The matrices are computed

$$\underline{\overline{Fn}} = \begin{bmatrix} \overline{0} & \overline{\delta}_3 & \overline{0} & -\overline{\delta}_3 \\ -\overline{\delta}_2 & \overline{0} & \overline{\delta}_2 & \overline{0} \\ -\overline{\delta}_3 * \overline{hs} & -\overline{\delta}_3 * \overline{hc} & -\overline{\delta}_3 * \overline{sn} & \overline{\delta}_3 * \overline{cs} \\ \overline{\delta}_2 * \overline{hc} & \overline{\delta}_2 * \overline{hs} & -\overline{\delta}_2 * \overline{cs} & -\overline{\delta}_2 * \overline{sn} \end{bmatrix},$$

$$\underline{\overline{Un}} = \begin{bmatrix} \overline{\delta}_0 & \overline{0} & \overline{\delta}_0 & \overline{0} \\ \overline{\delta}_0 & \overline{\delta}_1 & \overline{\delta}_0 & \overline{\delta}_1 \\ \overline{hc} & \overline{hs} & \overline{cs} & \overline{sn} \\ \overline{\delta}_1 * \overline{hs} & \overline{\delta}_1 * \overline{hc} & -\overline{\delta}_1 * \overline{sn} & \overline{\delta}_1 * \overline{cs} \end{bmatrix}.$$

Since Gauss elimination is usually available for left matrix division, we compute

$$\underline{\overline{Sn}} = \underline{\overline{Un}}^T \backslash \underline{\overline{F}}^T, \tag{49}$$

noting that $\underline{\overline{Sn}}$ is symmetric.

The convolution Gauss elimination is a copy of the number Gauss elimination with the four arithmetic number operations replaced by convolution number operations. The pivot search in the Gauss routine must be extended to search the pivot element in the current whole lower matrix slices. If no nonzero pivot element is found in the first slice, $r = 0$, then the search must go on in the second slice, etc. In the present calculation, the pivot element is found in slices $r = 0, r = 1, r = 2$,and $r = 3$ in the four successive elimination steps. Of course the following step of dividing by the pivot element must be compatible otherwise no convolution solution is possible, noting that leading Laurent series terms are excluded. In fact, the matrix $\underline{\overline{Un}}^{-1}$ cannot be computed because it is convolution singular, but the division in equation (49) can, due to the inclusion of compatible division in the convolution arithmetic routines.

Another notable feature occurs here. The determinant of $\underline{\overline{Un}}$ is a convolution number with leading element position $l = 6$ (counted from zero). Therefore, the compatible division of some matrix elements with highest element position m will produce a convolution number with highest element position at $m - 6$. If the analysis should retain system matrix slices up to ω^4, i.e., up to γ^8, then the convolution numbers in the matrix must be of length $n = 14$. Also, in the final result only every fourth slice is nonzero. Additional analytic manipulation can be carried out to avoid the many empty slices, but the purpose of the illustration is to show that successful computation can be done without special analytic treatment.

The computed convolution system matrix can be compared with the Taylor expansion of the analytical system matrix obtained by the same elimination process done analytically and simplifying with known hyperbolic and trigonometric identities,

$$\underline{\overline{S}} = EI \begin{bmatrix} \beta^3 \dfrac{\sinh\gamma\cos\gamma + \cosh\gamma\sin\gamma}{1 - \cosh\gamma\cos\gamma} & \beta^2 \dfrac{\sinh\gamma\sin\gamma}{1 - \cosh\gamma\cos\gamma} \\ \beta^2 \dfrac{\sinh\gamma\sin\gamma}{1 - \cosh\gamma\cos\gamma} & \beta \dfrac{\cosh\gamma\sin\gamma - \sinh\gamma\cos\gamma}{1 - \cosh\gamma\cos\gamma} \\ \beta^3 \dfrac{-\sinh\gamma - \sin\gamma}{1 - \cosh\gamma\cos\gamma} & \beta^2 \dfrac{\cos\gamma - \cosh\gamma}{1 - \cosh\gamma\cos\gamma} \\ \beta^3 \dfrac{\cosh\gamma - \cos\gamma}{1 - \cosh\gamma\cos\gamma} & \beta \dfrac{\sinh\gamma - \sin\gamma}{1 - \cosh\gamma\cos\gamma} \end{bmatrix}$$

$$\begin{bmatrix} \beta^3\dfrac{-\sinh\gamma-\sin\gamma}{1-\cosh\gamma\cos\gamma} & \beta^3\dfrac{\cosh\gamma-\cos\gamma}{1-\cosh\gamma\cos\gamma} \\ \beta^2\dfrac{\cos\gamma-\cosh\gamma}{1-\cosh\gamma\cos\gamma} & \beta\dfrac{\sinh\gamma-\sin\gamma}{1-\cosh\gamma\cos\gamma} \\ \beta\dfrac{\cosh\gamma\sin\gamma+\sinh\gamma\cos\gamma}{1-\cosh\gamma\cos\gamma} & \beta^2\dfrac{-\sinh\gamma\sin\gamma}{1-\cosh\gamma\cos\gamma} \\ \beta^2\dfrac{-\sinh\gamma\sin\gamma}{1-\cosh\gamma\cos\gamma} & \beta\dfrac{\cosh\gamma\sin\gamma-\sinh\gamma\cos\gamma}{1-\cosh\gamma\cos\gamma} \end{bmatrix}.$$

The inverse of this is given as "receptance" in [35]. Assuming that computers are now readily available, the analytical is much more difficult to obtain and must still be expanded with much analysis to obtain the numerical result that was obtained directly with convolution matrix algebra.

The actual numeric values of the nonzero slices $r = 0, r = 4, r = 8$ and the two pointer slices, $r = 15$ containing l, and $r = 16$ containing m, are given in Appendix 2. Note that the limiting length of the convolution numbers is due to row three, where the pointer $m = 8$.

The first two slices are the familiar stiffness and consistent mass matrices, [32,38,39]. The values of the third are the same as given in [32], noting that, according to the approach, there the third slice consists of $\mathbf{k}_4 - \mathbf{m}_2$ which turns out to be $\mathbf{m}_2$. The symbolic variable coefficients there are obtained by transforming $\gamma^4 = \dfrac{\omega^2\rho A l^4}{EI}$, $\gamma^8 = \dfrac{\omega^4(\rho A)^2 l^8}{(EI)^2}$. Again, it is clear that the higher matrix slices can hardly be called either stiffness or mass matrices any more. If any eigenvalues of this system matrix are computed, it must be remembered again that these are only valid below the first singularity at $\gamma = 4.73004\ldots$ which is the zero of the determinant.

4.3. Rational Convolution Number

For small vibration systems, it seems feasible to compute the condensed system matrix or even the inverse without loss of the singularities by using rational convolution numbers so that the result of division is written in the rational form of one polynomial divided by another, $p(x)/q(x) = \underline{X} \cdot \overline{p} \,/\, \overline{q}$. The arithmetic is then exact for all finite values of x within machine accuracy because the limit by any radius of convergence of Taylor series does not apply. If all the divisions that occur in the matrix division or inversion, i.e., in the Gauss decomposition and the backsubstitution, then the result is a matrix with rational numbers, but particularly with a common denominator which is the determinant of the matrix. This follows from the analytic expression of the inverse, now applied to convolution algebra,

$$\left[\underline{\overline{\overline{S}}}\right]^{-1} = \frac{adj\left[\underline{\overline{\overline{S}}}\right]}{|\underline{\overline{\overline{S}}}|} \equiv \frac{\underline{\overline{\overline{S}}}^{A}}{\overline{d}_S}, \tag{50}$$

where $\underline{\overline{\overline{S}}}^{A}$ is the adjoint convolution matrix, and $\overline{d}_S$ is the convolution determinant. The representation of a rational convolution matrix is therefore a convolution matrix, the numerator, and a single convolution number, the denominator. After the discussion of the inverse of a convolution matrix, it is clear that every rational convolution matrix has a convolution number as denominator, which in the case of the matrix $\underline{\overline{\overline{S}}}$ is $\overline{\delta}_0$. Similarly, every conformable convolution matrix division can be brought into the form of a convolution matrix as numerator and a convolution number as denominator. We denote any rational convolution matrix $\underline{\overline{\overline{H}}}$ in fraction form by

$$\underline{\overline{\overline{H}}} = \underline{\overline{\overline{H}}}^{P} \,/\, \overline{q}_H,$$

where the numerator $\underline{\overline{\overline{H}}}^{P}$ consists of finite convolution numbers and the denominator $\overline{q}_H$ is a single finite convolution number. Therefore, if H is the name of the inverse of S in equation (36)

then $\underline{\underline{\overline{S}}}^{A} = \underline{\underline{\overline{H}}}^{P}$ and $\overline{d}_S = \overline{q}_H$.

The computation of the adjoint by $n \times n$ subdeterminants becomes prohibitive for any $n > 4$. An effective method to determine the adjoint matrix and the determinant is the well-documented Gauss elimination and backsubstitution for rational numbers, also called divisionless Gauss elimination [34]. The method is applicable to a square matrix with any length convolution numbers, and we will show later how these are produced starting from the conventional form $\underline{\overline{S}}(\omega) = \underline{\overline{K}} + \hat{\imath}\omega\underline{\overline{C}} - \omega^2\underline{\overline{M}}$. Some of the features that occur when the method is applied to convolution numbers will now be shown. We only consider convolution matrices with equal convolution number lengths r.

During the Gauss decomposition, a lower left and upper right matrix is created in which the untruncated convolution numbers become longer with each elimination step until the lowest right element is the determinant which has length $n \times r$. Unless special provision for unequal lengths in the matrix is made, $n \times n \times n \times r$ storage positions are required.

It is interesting to note that in this method a division is required because a common factor occurs only after addition of elements. Consider the following matrix elements after the elimination of the first two columns of a convolution matrix $\underline{\underline{\overline{a}}}$ has already been made, the uncompleted elements being denoted by $\overline{a}_{ij}$, the completed by $\overline{b}_{ij}$, and the completed diagonals by $\overline{d}_{ii}$.

$$\begin{array}{|llll|}
\hline
\overline{d}_{11} & \cdots & & \overline{b}_{15} \\
\overline{b}_{21} & \overline{d}_{22} & \cdots & \overline{b}_{25} \\
\overline{b}_{31} & \overline{b}_{32} & \cdots & \overline{a}_{35} \\
\vdots & & & \\
\hline
\end{array}$$

The next elimination step is to replace the elements in the third row, e.g., $\overline{a}_{35}$, by

$$\overline{b}_{35} = \frac{\overline{d}_{22} * \overline{a}_{35} - \overline{b}_{32} * \overline{a}_{25}}{\overline{d}_{11}}.$$

The factor $\overline{d}_{11}$ can only be cancelled after the subtraction operation in the numerator. If during the course of the decomposition the new element occupies s positions and the denominator t, then the numerator occupies $s + t$ positions which is more than provided for, particularly for the last element $\overline{d}_{nn}$, which occupies $n \times r$ positions. But fortunately, according to Theorem 4 in Section 11, only the s positions of the numerator need to be computed, i.e., in no case more than the maximum length of $n \times r$ needs to be provided for the matrix elements. The general form of the elimination step above is

$$\overline{b}_{ij} = \frac{\overline{d}_{kk} * \overline{a}_{ij} - \overline{b}_{ik} * \overline{a}_{kj}}{\overline{d}_{(k-1),(k-1)}}, \tag{51}$$

which must be performed from $k = 1$ to n, $\overline{d}_{00}$ taken as 1. This is only one of possible sequences which has been chosen because it requires the minimum number of extraction of indexed numbers, but mainly because it allows a complete convolution pivot search.

Consider now the backsubstitution steps, e.g., when the 4×4 matrix has been decomposed as shown below, with the decomposed right elements $\overline{r}_i$ and the unknown convolution vector elements $\overline{x}_j$.

$$\begin{array}{cccc|c} \overline{x}_1 & \overline{x}_2 & \overline{x}_3 & \overline{x}_4 & \\ \hline \overline{d}_{11} & \overline{b}_{12} & \overline{b}_{13} & \overline{b}_{14} & \overline{r}_1 \\ & \overline{d}_{22} & \overline{b}_{23} & \overline{b}_{24} & \overline{r}_2 \\ & & \overline{d}_{33} & \overline{b}_{34} & \overline{r}_3 \\ & & & \overline{d}_{44} & \overline{r}_4 \end{array}$$

Starting backsubstitution from the left,

$$\overline{x}_4 = \frac{\overline{r}_4}{\overline{d}_{44}} = \frac{\overline{r}_4}{\overline{d}_A}.$$

Only $\overline{r}_4$ is stored in the position allocated for $\overline{x}_4$ while $\overline{d}_{44} = \overline{d}_A$, which is the convolution determinant, is stored in a separate position which is common for the solution vector or matrix. The next backsubstitution is

$$\begin{aligned} \overline{x}_3 &= \frac{\overline{r}_3 - \overline{b}_{34} * \overline{x}_4}{\overline{d}_{33}} \\ &= \frac{\overline{r}_3 * \overline{d}_{44} - \overline{b}_{34} * \overline{r}_4}{\overline{d}_{33}} / \overline{d}_A. \end{aligned} \tag{52}$$

Here again, in equation (52), $\overline{d}_{33}$ is a factor and the division produces a convolution number of length $n \times r$, so that the numerator can be truncated to the same length. The numerator after division in equation (52) is stored in the position allocated for $\overline{x}_3$. Continuing for the elements $\overline{r}_2 \rightarrow \overline{x}_2$, $\overline{r}_1 \rightarrow \overline{x}_1$, with the general formula

$$\overline{r}_i = \frac{\overline{r}_i * \overline{d}_{44} - \sum_{l=i+1}^{n} \overline{b}_{il} * \overline{r}_l}{\overline{d}_{ii}}, \tag{53}$$

$$\overline{x}_i = \frac{\overline{r}_i}{\overline{d}_A}. \tag{54}$$

The denominator is always $\overline{d}_A$ and the numerator always an exact convolution number.

In all the divisions, as described above, the pointer m is shifted back after the division to the correct length of the result but, because of numerical truncation errors, small numbers may remain in the higher elements which may disturb the correct setting of m. To avoid this, a cleanup operation may be programmed after each division operation. The theoretical length is known from the m-pointers, although the complex numbers make this slightly more complicated, and the higher elements are filled with zeros.

The divisionless Gauss elimination and backsubstitution gives the result which is described analytically by the form of equation (50), from which it can be seen that any square rational convolution matrix can be inverted, there is no limitation due to singular matrix slices, but it is not possible for convolution singular matrices where the determinant is $\overline{0}$. The reason is that in such a case a diagonal element $\overline{d}_{ii}$ will become $\overline{0}$ and the division step is not possible. In any case, because a division step is involved, a pivot search as in Section 4.2 must be included for the same reasons as described there. With this routine, system matrices that have mass or damping matrices but no stiffness matrix can be still inverted.

Also, the divisionless Gauss elimination without the backsubstitution step can be used to determine the exact convolution determinant. From this again, the zeros of the characteristic polynomial can be determined by a polynomial root solving routine. This means that eigenvalues can be determined for unconventional matrices, i.e., consisting of more than two or three slices. Of course, this application is still limited by the fact that polynomial coefficients may become too large very quickly to be handled by ordinary computer number length.

The rational form of the condensed system matrix of equation (43) will be

$$\begin{aligned}\overline{\overline{S11}} &= \overline{\overline{S}}_{11} - \overline{\overline{S}}_{12} * \overline{\overline{S}}_{22}^{A} * \overline{\overline{S}}_{21} / \overline{d}_{22} \\ &= \left[\overline{\overline{S}}_{11} * \overline{d}_{22} - \overline{\overline{S}}_{12} * \overline{\overline{S}}_{22}^{A} * \overline{\overline{S}}_{21} \right] / \overline{d}_{22}, \end{aligned} \tag{55}$$

which is now of the general rational form. In fact the beam matrix of equation (46) in Section 4.2 is exactly such a condensed matrix, albeit infinitely, to the forces at the end points.

Let a be the dimension of the partition 11 and $b = n - a$ the length of the partition 22 , then the lengths of convolution numbers in the different terms are r in $\overline{\overline{S}}_{11}$, br in $\overline{d}_{22}$, r in $\overline{\overline{S}}_{12}$, and $\overline{\overline{S}}_{21}$, and $(b-1)r$ in $\overline{\overline{S}}_{22}^{A}$. The first term $\overline{\overline{S}}_{11} * \overline{d}_{22}$ is therefore a matrix with convolution numbers of length $r + br = r(1+b)$, and the second term $\overline{\overline{S}}_{12} * \overline{\overline{S}}_{22}^{A} * \overline{\overline{S}}_{21}$ a matrix with convolution numbers of length $r + (b-1)r + r = r(1+b)$, which means that convolution numbers of same lengths are added. The rational form consists of a convolution matrix of length $r(1+b)$ and a convolution denominator of length br. The convolution number length of the original system matrix consisting of stiffness, damping and mass matrices is $r = 2$.

Consider the inverse of this general rational convolution matrix

$$\begin{aligned}\left[\overline{\overline{S}} \right]^{-1} = \left[\overline{\overline{S11}}^{P} / \overline{d}_{22} \right]^{-1} = \overline{d}_{22} * \left[\overline{\overline{S11}}^{P} \right]^{-1} &= \overline{d}_{22} * \left[\overline{\overline{S11}}^{P} \right]^{A} / \overline{d}(S11^{P}) \\ &\equiv \overline{\overline{H11}}^{P} / \overline{d}(S11^{P}). \end{aligned}$$

The length of the convolution determinant $\overline{d}(S11^{P})$ is $ar(1+b) = (a+b)r$, and the length of the convolution matrix $\overline{\overline{H11}}^{P}$ is $(a-1)r(1+b) = (a+b-1)r$. Consider, in comparison, the inverse of the whole system matrix

$$\left[\overline{\overline{S}} \right]^{-1} = \overline{\overline{S}}^{A} / \overline{d}_{S} \equiv \overline{\overline{H}}^{P} / \overline{q}_{H}, \tag{56}$$

where $\overline{\overline{H}}$ is the Taylor transform of $\underline{H}(\omega)$ which is known as the transfer function matrix, being a matrix of transfer functions which each is a rational polynomial; it is also sometimes called the frequency response function matrix.

The length of the convolution determinant is $n \times n \times 2$ and of the convolution matrix $(n-1) \times 2$. It can be shown that the inverse matrix $\left[\overline{S11}(\omega) \right]^{-1}$ is identical to the partition $\overline{H}_{11}$ and from the above it can be seen that the length of the corresponding convolution matrices coincide.

The same operations apply to linear control theory where the corresponding matrix is $\overline{G}(s) = \overline{G}(i\omega) = \overline{H}(\omega)$.

EXAMPLE 1. Finally, as illustration, the inverse of a system matrix of a simple 4 DOF mass-damper-spring system shown in Figure 3 is computed.

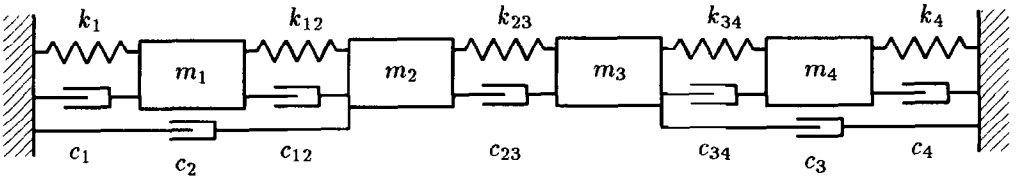


Figure 3. 4 DOF spring mass damper system.

The physical parameters are (units not given):

$$\begin{aligned} &m_1 = 2, \quad m_2 = 1, \quad m_3 = 1, \quad m_4 = 2 \\ &k_1 = 100, \quad k_{12} = 100, \quad k_{23} = 100, \quad k_{34} = 100, \quad k_4 = 100 \\ &c_1 = 0.1, \quad c_{12} = 0.1, \quad c_2 = 0.01, \quad c_{23} = 0.1, \quad c_3 = 0.01, \quad c_{34} = 0.1, \quad c_4 = 0.1. \end{aligned}$$

The system matrices are calculated from these, [31,35],

$$\underline{\overline{K}} = \begin{bmatrix} 200 & -100 & 0 & 0 \\ -100 & 200 & -100 & 0 \\ 0 & -100 & 200 & -100 \\ 0 & 0 & -100 & 200 \end{bmatrix}, \quad \underline{\overline{C}} = \begin{bmatrix} 0.2 & -0.1 & 0 & 0 \\ -0.1 & 0.21 & -0.1 & 0 \\ 0 & -0.1 & 0.21 & -0.1 \\ 0 & 0 & -0.1 & 0.2 \end{bmatrix},$$

$$\underline{\overline{M}} = \begin{bmatrix} 2.0 & & & \\ & 1.0 & & 0 \\ & & 1.0 & \\ & 0 & & 2.0 \end{bmatrix}.$$

The system matrix and its Taylor transform is defined by

$$\underline{\overline{S}}(\omega) = \underline{\overline{K}} + \hat{\imath}\omega\underline{\overline{C}} - \omega^2\underline{\overline{M}} = \underline{W} \cdot \left[\underline{\overline{K}}, \hat{\imath}\underline{\overline{C}}, -\underline{\overline{M}}\right] \equiv \underline{W} \cdot \underline{\overline{\overline{S}}}.$$

The inverse $\underline{\overline{\overline{H}}}$ is computed as described above, the numerical results are given in Appendix 3. In the Figure 4(a) below is the result $\underline{\overline{H}}(\omega)$ obtained by evaluating each polynomial H_{ij} and the determinant $p(\omega)$ for each value of ω shown graphically to a frequency of 24 rad/s, in a view resembling the three dimensions of the transfer function matrix. The plots show the amplitude values on a log scale with the familiar resonance and anti-resonance peaks. In Figure 4(b), the single transfer function $H_{11}(\omega)$ is shown for the successive condensed system matrix $S11(\omega)$, and truncated to three, two and one dimensions, revealing the truncation error.

The convolution method of determining $\underline{\overline{H}}(\omega)$ needs large computer storage and much computation time for evaluating the convolution matrix $\underline{\overline{\overline{H}}}$. Once this has been done, the computation of $\underline{\overline{H}}(\omega)$ needs approx. $n{\times}r$ multiplications for each element, therefore, $n{\times}n{\times}n{\times}r$ multiplications.

An alternative method is to evaluate the system matrix $\underline{\overline{S}}(\omega)$ and then inverting the matrix for each ω. This method requires much less computer storage. The computation consists of r multiplications for each element, therefore, $n{\times}n{\times}r$ multiplications and then of the order n^3 multiplications for the inversion. Total multiplications are therefore approx. $n{\times}n{\times}(n+r)$. Obviously, the approach to determine the values of $\underline{\overline{H}}(\omega)$ from the convolution matrix $\underline{\overline{\overline{H}}}$ is not efficient at all. The determination of $\underline{\overline{\overline{H}}}$ must be seen for analytic purposes only.

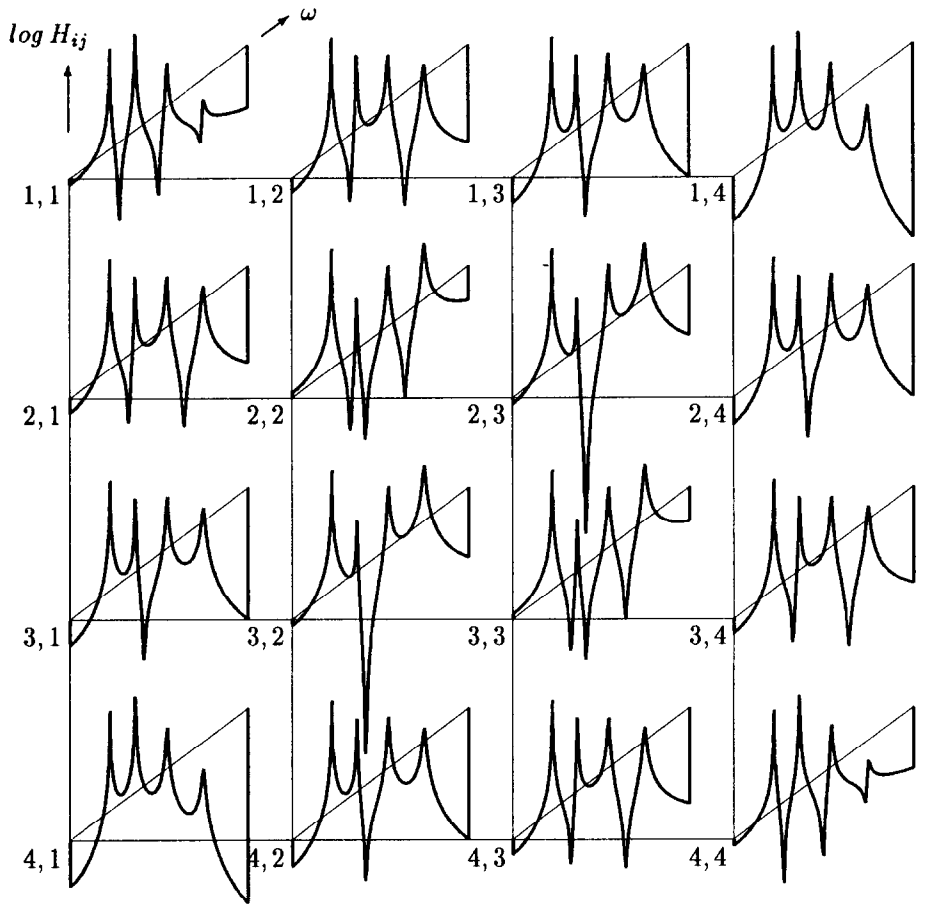


Figure 4a. Transfer function matrix $\underline{\overline{H}}(\omega)$.

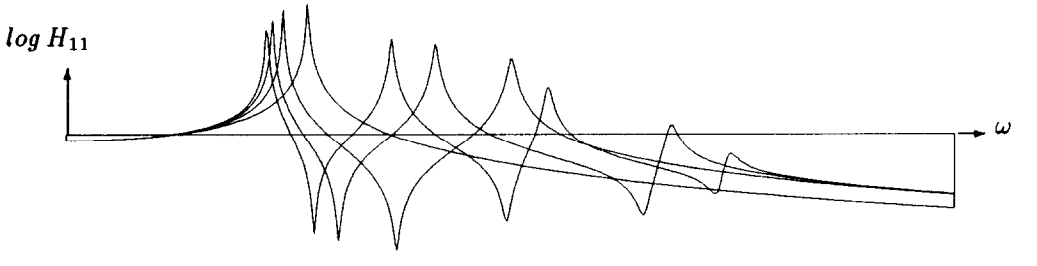


Figure 4b. Transfer function $H_{11}(\omega)$.

Table 1. 4×4 DOF System convolution matrix determinant by different methods.

	Exact within 7 digits		Divisionless Elimination		Gauss Elimination	
r	real	imag.	real	imag.	real	imag.
0	5E+08	0	5E+08	0	5E+08	0
1	0	2120000	0	2120000	0	2120000
2	−2.800336E+07	0	−2.800337E+07	0	−2.800336E+07	0
3	0	−87602.37	0	−87602.35	0	−87602.37
4	440091.3	0	440091.2	0	440091.2	0
5	0	912.0317	0	912.0316	0	912.0316
6	−2400.472	0	−2400.473	0	−2400.472	0
7	0	−2.48	0	−2.480003	0	−2.479999
8	4	0	4.000006	0	3.999998	0
l	0	1	0	1	0	1
m	8	7	8	7	8	7

Both methods are linear. The alternative to determine the eigenvalues and eigenvectors and then computing the convolution transfer function matrix for each ω by a simple left and right matrix multiplication of a diagonal convolution matrix is only available for system matrices with two slices. The system with the three slices as used above must then be tranformeed to a system with two slices with double the dimension, [31,33].

4.4. Alternative Method

Division by a rational convolution matrix, or particularly the inverse, can be done by ordinary convolution Gauss elimination. To be specific consider the same example of the 4×4 system convolution matrix $\underline{\overline{\overline{K}}}, \hat{\imath}\,\underline{\overline{\overline{C}}}, -\underline{\overline{\overline{M}}}$ as in the example above. We only need to allow for a length of 3×2 = 6 convolution number for the inverse, but for simplicity of having all convolution numbers the same length we have allowed a length of 4×2 = 8 which is the length of the convolution determinant. Now we proceed with the ordinary convolution Gauss elimination and backsubstitution as was described briefly in Section 4. This requires only half of the amount of multiplication steps as the divisionless Gauss elimination, but now the inverse convolution matrix $\underline{\overline{\overline{H}}}$ is filled with the truncated of the infinite convolution numbers corresponding to the Taylor expansions of the actual convolution numbers, as was the case in the rod and beam example of the previous section, except carried out to at least a length of $(n-1)\times r$. But now, according to Theorem 4 (see Section 11), the exact convolution numbers of the adjoint matrix can be recovered from the preliminary convolution matrix $\underline{\overline{\overline{H}}}$ as follows. After the forward decomposition, the determinant is computed as in ordinary number matrix algebra by the product of the diagonals, truncated at the exact theoretical length. Each element of the preliminary inverse convolution matrix $\underline{\overline{\overline{H}}}$ is now multiplied with the determinant and truncated at the correct length by the same cleanup operation as described in Section 4.3 above. It may come as a surprise that this computation is actually more accurate than the theoretically accurate in Section 4.3. The convolution determinant calculated by the two methods on a IBM PC with seven digits is compared in the Table 1 below with the exact within the same number of digits as calculated on the HP 9836 with 13 digits, checked with a double precision calculation on the IBM PC. The better accuracy is due to the lesser number of multiplications involved for each number.

In fact, by the convolution Gauss decomposition, the convolution determinant of a system matrix with any number of slices can be found, and then the eigenvalues by a polynomial root solver.

In summary, we conclude that the convolution matrix is a convenient semi-analytic tool, but needs a large amount of computer storage. But it has also the inherent difficulty that polynomial

coefficients become very large very quickly and the method could be better exploited when the next generation of computers with variable digital number length becomes available.

5. ALGEBRA OF THE CONVOLUTION NUMBER

Once the arithmetic of convolution numbers is defined the algebra follows. The algebra of convolution numbers is the algebra of sequences which is well-documented in [10], and the algebra of polynomials [13]. With the arithmetic routines described in Section 3 above an algebra of convolution numbers is established.

EXAMPLE 2. As an example of convolution algebra, we compute the conformal mapping of a circle on a hyperbolic profile by means of a series. To generate the hyperbolic profile in the z-plane, a mapping complex potential function is used

$$w_m = (z - \hat{c})^2 - \hat{\imath}\psi_0, \tag{57}$$

where $\hat{\imath}$ is the imaginary unit[6] and $\hat{c} = c_x + \hat{\imath} c_y$ is a complex number to be determined. The mapping potential lines and hyperbolic streamlines are shown in Figure 5(a), where at the stagnation point $z = \hat{c}$ the value of $w_m = 0 - \hat{\imath}\psi_0$. A hyperbolic profile is selected as shown by the marks. The streamline is defined by choosing the value of ψ_0 and the hyperbolic arc is defined by the potential values ϕ_1 and ϕ_2 at the two ends. A circle with diameter a is placed in the mapping flow resulting in a new potential flow in the ζ-plane as shown in Figure 5(b), which is given by the complex potential function from the circle theorem of Milne-Thomson, [40],

$$w_m = \left(\zeta - \hat{c}\right)^2 + \left(\frac{a^2}{\zeta} - \overset{\times}{\hat{c}}\right)^2 + \phi_c, \tag{58}$$

where $\overset{\times}{\hat{c}}$ is the complex conjugate of $\hat{c}$. The four constants c_x, c_y, a, ϕ_c are determined iteratively by the four conditions that the mapping potential values at the circle stagnation points are ϕ_1 and ϕ_2, respectively, and the complex mapping potential at the field stagnation point has the value $w_m = 0 - \hat{\imath}\psi_0$. The one to one mapping is given implicitely by equations (57) and (58), producing the implicit mapping equation after rearranging

$$z^2 - 2z\hat{c} = \zeta^2 - 2\zeta\hat{c}\ \overset{\times}{\hat{c}}^2 + \phi_c + \hat{\imath}\psi_0 - \frac{2a^2\ \overset{\times}{\hat{c}}}{\zeta} + \frac{a^4}{\zeta^2}. \tag{59}$$

The mapping can be verified by the mapping stream and potential lines in the Figures 5(a) and 5(b). The posed problem is to find the explicit mapping function in the form of a half infinite Laurent series

$$z = \zeta + z_0 + \frac{z_{-1}}{\zeta} + \frac{z_{-2}}{\zeta^2} + \cdots, \tag{60}$$

which converges outside the circle in the ζ-plane. We don't discuss here why an explicit mapping function would be required. To use convolution numbers based on Taylor series, we transform to a new variable $p = \dfrac{1}{\zeta}$ so that equation (60) becomes

$$\begin{aligned} z &= \zeta + u_0 + u_1 p + u_2 p^2 + \cdots \\ &\equiv \zeta + u(p), \end{aligned} \tag{61}$$

[6]The symbol $\hat{\imath}$ for the imaginary unit was introduced in [25]. We use generally $\hat{c}$ for a complex number, default plain c, and $\overset{\times}{\hat{c}}$ for the complex conjugate, which are distinctive symbols that cannot be confused with real numers or vectors.

where $u(p)$ is the Taylor series to be found. Substituting equation (61) into (59), cancelling the highest powers of ζ and multiplying by p to avoid negative powers,[7] we arrive at the equation

$$2u(p) + pu^2(p) - 2\hat{c}\,pu(p) = \left(\overset{\times}{c}^{\,2} + \phi_c + \hat{\imath}\,\psi_0\right) p - (2a^2\,\overset{\times}{c})p^2 + a^4p^3. \tag{62}$$

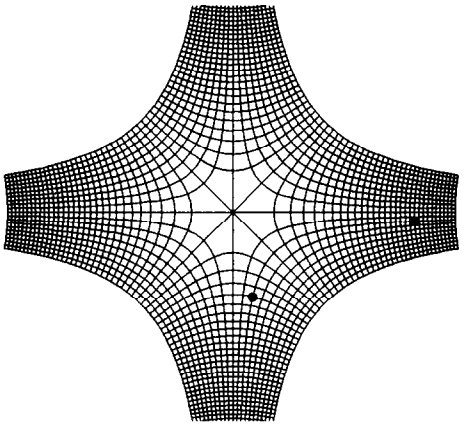


Figure 5a. Hyperbolic mapping lines.

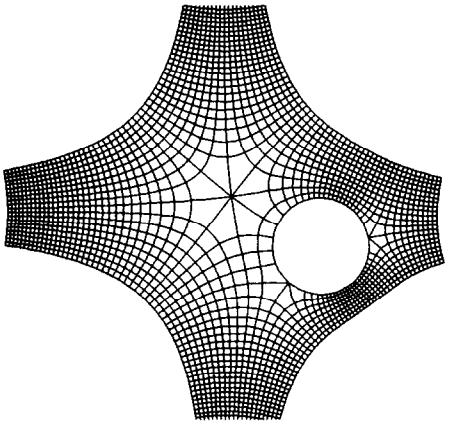


Figure 5b. Circle inserted.

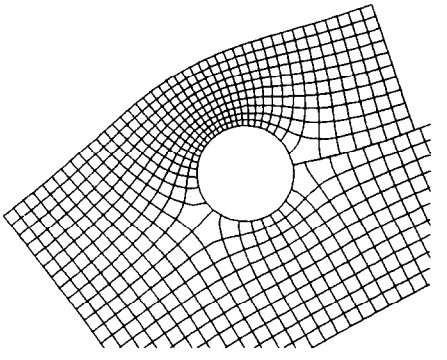


Figure 5c. Circle stream lines.

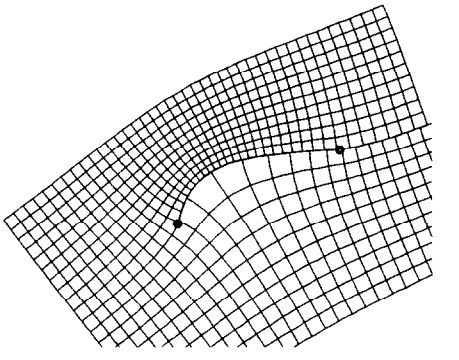


Figure 5d. Hyperbolic stream lines.

Equation (62) is transformed to a convolution number equation by applying the transformation equation (4)

$$2\overline{u} + \overline{\delta}_1 * \overline{u}^{*2} - 2\hat{c}\;\overline{\delta}_1 * \overline{u} = \overline{b}, \tag{63}$$

where the convolution number

$$\overline{b} \;=\; \left\{0,\; \left(\overset{\times}{c}^{\,2} + \phi_c + \hat{\imath}\,\psi_0\right),\; -2a^2\,\overset{\times}{c},\; a^4\right\}. \tag{64}$$

The quadratic equation can be solved explicitly by the formula, using the five arithmetic routines,

$$\overline{u} = \left(-\left(1 - \hat{c}\;\overline{\delta}_1\right) \pm \sqrt{\left(1 - \hat{c}\;\overline{\delta}_1\right)^2 + \overline{b} * \overline{\delta}_1}\;\right) / \overline{\delta}_1. \tag{65}$$

[7]Note that by this means we avoid Laurent series terms.

The numerical details of this example are given and discussed in the Appendix 4. The result is shown graphically by plotting the uniform flow stream and potential lines around the circle in the ζ-plane at the ideal incidence in Figure 5(c) and the flow about the hyperbolic profile in Figure 5(d), for which the actual computed series of the mapping function of equation (61) was used.

6. SCALAR FUNCTION OF A CONVOLUTION NUMBER

A scalar function of a variable is defined as a function which only uses the previously described arithmetic operations on the variable, where scalar refers to the function, not to the value. This is analog to the definition of a real function of a complex variable. With the definition of the operations with a scalar, a scalar function of a convolution number is then a scalar function of a variable, where the variable is a convolution number, scalar coefficients remain scalar coefficients. Consider a function of the variable u, e.g., $f(u) \equiv a + bu + cu^2$, and now let u be a function of x, i.e., $u(x)$, expressed as a Taylor series by equation (4), $u(x) = \underline{X} \cdot \overline{u}$ which defines the convolution number $\overline{u}$. The function $f(u)$ becomes a function of x, $f(u(x))$, which is obtained by substituting the series in the expression of the function, and which is expressed in terms of convolution algebra as $f(u(x)) = \underline{X} \cdot (a + b\overline{u} + c\overline{u}^{*2})$. These ideas are stated by the following lemma.

6.1.

LEMMA 1. *If the Taylor transform of a function $u(x)$ is*

$$u = \underline{X} \cdot \overline{u},$$

then a scalar function of u is transformed as

$$f(u) = \underline{X} \cdot f(\overline{u}) \equiv \underline{X} \cdot \overline{f},$$

where $f(\overline{u})$ is the same scalar function of the convolution number $\overline{u}$ as $f(u)$ of the number u with the $+$, $-$, $.$, $/$ operations on a scalar variable replaced by the $+$, $-$, $*$, $/$ operations on a convolution number.

This lemma can be interpreted as being a definition of the algebraic processes that must be performed on convolution numbers to be transforms of the algebraic processes on the Taylor series. It is the same theorem as for every linear transform, here stated for convolution numbers. Whenever a scalar function $f(u)$ is given, and u again is a function of x , $u(x)$ which can be expanded in a Taylor series, Lemma 1 states that the Taylor expansion of $f(u(x))$ has the coefficients $f(\overline{u})$ which can be expressed by a new convolution number $\overline{f}$. The radius of convergence of the series $\underline{X} \cdot \overline{f}$ depends on both $u(x)$ and f and will be examined for each application separately.

This problem is called composition, meaning a function is substituted as argument into another function [12], and is used specifically for $f(u)$ being a polynomial [8]. We consider here any functions that are expressed as Taylor series, which is later extended to any function that is defined analytically. The decisive difference is that such functions do not occur in terms of their Taylor series at all but are evaluated directly in terms of convolution algebra, as in Example 2. We therefore apply the term composition for the substitution of a Taylor series for the argument of any analytical function, but use this term in the number domain only because by the very definition of a convolution function of a convolution number the transformed operation is the substitution of an argument into a function. Unfortunately, we cannot give the operations in the number domain and convolution domain exactly the same name as with multiplication because we don't want to change accepted useful terms in the number domain. We have to accept then that *composition in the number domain transforms to substitution in the convolution domain.*

7. CONVOLUTION VARIABLE

We use the isomorphism between function of numbers and functions of convolution numbers to examine how to distinguish between convolution constants and variables.

7.1. Convolution Constant

Let us consider a function $f(x)$, e.g., $\cos(x)$, and call this function $c(x)$. The Taylor transform of $c(x)$ is the convolution number given by the coefficients of the Taylor series represented by equation (4),

$$\overline{c} = \{1\ 0\ 0.5 \ldots\}.$$

Wherever the function $c(x)$ appears in a functional relation or equation, it is always transformed to the same convolution number $\overline{c}$. In terms of convolution algebra, we call this a *convolution constant*; i.e., any convolution number representing a predetermined function of a continuous number variable is a constant of convolution number algebra. Note that we have to distinguish between the continuous variable x, which is a number variable, and the convolution constant $\overline{c}$. This definition will become clearer when we compare it with a convolution variable.

7.2. Convolution Variable

In the numerical, Example 2 of Section 5.1, the convolution number $\overline{u}$ was not known before the solution. This corresponds to a complete function $u(p)$ which was unknown. If equation (62) had different parameters of the hyperbola, $u(p)$ would be a different function and $\overline{u}$ a different convolution number. Also, if the problem would be solved by an iteration method, many convolution numbers $\overline{u}$ would have occurred, each representing a different function $u(p)$. Such a convolution number that can vary for a number of reasons is called a *convolution variable* of convolution number algebra. Note that we have to distinguish between the continuous variable x, which is a number variable, and the convolution variable $\overline{u}$. This definition will become clearer when we discuss the convolution differential.

7.3. Convolution Function

A scalar function of a convolution number was defined with the example of number constants which transformed to special convolution constants. Consider now the convolution constants $\overline{a}, \overline{b}, \overline{c}$ and the convolution variable $\overline{u}$ in the same example and construct analog to the scalar function of a number variable in the number domain, a function in the convolution domain

$$\overline{f} = \overline{a} + \overline{b} * \overline{u} + \overline{c} * \overline{u}^{*2}. \tag{66}$$

We call this a *convolution function* $\overline{f}(\overline{u})$ *of a convolution variable* $\overline{u}$. The corresponding function in the number domain, i.e., the inverse Taylor transform of equation (66), is

$$f(u) = a(x) + b(x).u(x) + c(x).u^2(x). \tag{67}$$

We note that in the number domain it is not obvious which functions are coefficients and which are functions, because it is actually merely a matter of definition, the fact that $u(x)$ occurs repeated is not decisive. Actually, the same ambiguity may occur with scalar functions when derivatives w.r.t. coefficients are considered, i.e., scalar coefficients are considered as variables in certain situations. We are, however, used to the occurrence of functions $a(x), b(x), c(x)$ being called coefficients in differential equations, e.g., $L(u(x)) = a(x) + b(x).u'(x) + c(x).u''(x)$. It is in this sense that functions in the number domain transform to constants or variables in the

convolution domain. We denote the corresponding convolution function of a convolution variable by $\overline{f}(\overline{u})$ to distinguish them from the scalar function of a convolution variable denoted by $f(\overline{u})$. Obviously, $\overline{f}(\overline{u})$ is the more general convolution function and includes the scalar convolution function $f(\overline{u})$. To distinguish, we will call functions in the number domain that have functions like generalized coefficients as functional functions.[8] The customary notation for such functions is $f(u(x), x)$, [29], to show that they are implicit and explicit functions of the variable x. But this distinction cannot be transformed like that into the convolution number domain because it would destroy the isomorphism. A consistent notation would be $f(u(x), c_i(x))$ for the functional function, corresponding to a scalar function notation of $f(u(x), c_i)$, and the convolution function notation of $\overline{f}(\overline{u}, \overline{c}_i)$ and $f(\overline{u}, c_i)$, respectively. By this elaboration, we hope to prevent any confusion if we use the inconsistent but convenient notations $f(u(x), x)$ and $\overline{f}(\overline{u})$. We have used the term coefficient here which suited the notation, generally they may occur as parameters which are not necessarily coefficients. Finally, using the comparison with complex functions again, the scalar function of a convolution variable is the analog of the real function of a complex variable and the convolution function of a convolution variable is the analog of the complex function of a complex variable.

From the convolution algebra as described it is clear that the convolution function of equation (66) is the Taylor transform of the function of equation (67)

$$a(x) + b(x).u(x) + c(x).u^2(x) = \underline{X} \cdot \left(\overline{a} + \overline{b} * \overline{u} + \overline{c} * \overline{u}^{*2} \right).$$

Note that the composition problem is now generalized to functional functions. We extend now Lemma 1 which was stated for scalar functions to a theorem.

7.4.

THEOREM 1. *If the Taylor transform of a function $u(x)$ is*

$$u = \underline{X} \cdot \overline{u},$$

then a functional function of u is transformed by

$$f(u) = \underline{X} \cdot \overline{f}(\overline{u}) \equiv \underline{X} \cdot \overline{f},$$

where $f(\overline{u})$ is the same convolution function of the convolution number $\overline{u}$ as $f(u)$ of the number u with the algebraic operations and variable coefficients on a scalar variable replaced by the transformed operations and convolution coefficients of a convolution variable.

The same comments as for Lemma 1 apply, specifically that all operations on a number variable that produce a function of the variable are transformed by Theorem 1 into convolution operations of a convolution variable. It also includes composition of a chain of functions, the last function being a Taylor series, which transforms to substitution, the last function transforming to the convolution number. In this theorem, it is implied that $f(u)$ includes $f(u(x), x)$ but as far as the theorem is concerned there is no distinction between constants and variables. In view of the numerical applications that are our concern in this treatise, we must find the computational routines for convolution operations first before we can apply the theorem. This has been done for algebraic routines in the previous sections and demonstrated in Example 2 where the functional function of equation (62) was transformed to a convolution function in equation (63). We have yet to transform the analytic operations of differentiation and integration of number variables. We will find that to preserve the isomorphism, the latter do not transform according to the customary

[8]We hope that this will not lead to confusion with the functional of Variational Calculus, which is a scalar.

meanings, a feature which is already partly familiar from the well-known linear transforms. Also, to apply the theorem, the Taylor expansions of the functional coefficients must be known, but only numerically. To find these numbers in all cases efficiently is part of this treatise.

8. CONVOLUTION DIFFERENTIAL

Analog to the differential dx of a number variable, we define a convolution differential $d\overline{u}$ of the convolution variable $\overline{u}$. The differential $d\overline{u}$ consists of infinitely many independent differentials du_i

$$d\overline{u} = \{du_0\ du_1\ du_2 \ldots\}. \tag{68}$$

The transform of $d\overline{u}$ to the number domain is the variation of the function $u(x)$, in the sense of Variational Calculus [36],

$$\delta u(x) = \underline{X} \cdot d\overline{u}. \tag{69}$$

This transform clarifies further the distinction between a convolution constant and a convolution variable: obviously we cannot take a differential of a constant $\overline{c}$ which corresponds in the number domain to the variation of a fixed function like $\cos(x)$.

9. CONVOLUTION DERIVATIVE

The algebra of numbers is used to define a derivative, and therefore we use the isomorphism with convolution algebra to define a convolution derivative of a convolution function $\overline{f}(\overline{u})$ of a convolution variable $\overline{u}$ with the aid of the differential

$$d\overline{f} = \left(d\overline{f}/d\overline{u}\right) * d\overline{u}, \tag{70}$$

where the derivative again is a convolution function of the convolution variable $\overline{u}$, $d\overline{f}/d\overline{u} \equiv \overline{g}(\overline{u})$, and the dependent differential $d\overline{f}$ is defined by

$$d\overline{f} = \overline{f}\left(\overline{u} + d\overline{u}\right) - \overline{f}(\overline{u}).$$

The inverse transform of equation (70) is the variation

$$\delta f = \frac{df}{du}.\delta u,$$

in the customary notation of variational calculus if the function is only an implicit function $f(u(x))$, but it is better known as[9]

$$\delta f = \frac{\partial f}{du}.\delta u,$$

because the function is considered to be also an explicit function of x, $f(u(x), x)$. The convolution derivative of a convolution function is determined by the same rules as a derivative of a function in the number domain, using known formulas for known functions. Analog to the higher derivatives and their symbols for functions in the number domain

$$\frac{df}{du} = f' = g,$$

$$\frac{d}{dx}\left(\frac{df}{dx}\right) = \frac{d^2f}{dx^2} = f'' = g' = h,$$

[9]We use a revised partial differential notation, showing where a partial increment is divided by a total increment, [41].

we continue to define higher convolution derivatives, and symbols for derivatives of convolution functions

$$d\overline{f}/d\overline{u} = \overline{f}^{(*)} = \overline{g},$$
$$d/d\overline{u}\left(d\overline{f}/d\overline{u}\right) = d^2\overline{f}/d\overline{u}^{*2} = \overline{f}^{(**)} = \overline{g}^{(*)} = \overline{h}.$$

Note that these are not transforms of each other, $\overline{g}$ is not the transform of f'. To make the transform clearer the term *variational* was introduced for $\frac{\partial f}{du}$ and written $\frac{\delta f}{\delta u}$, [36]. With this definition, $\overline{g}$ is the transform of the variational of f, and $\overline{h}$ is the transform of the second variational of f.

Analog to the indefinite integral of a function in the number domain

$$\int g dx = f(x),$$

the convolution integral is

$$\int \overline{g}\, d\overline{u} = \overline{f}(\overline{u}).$$

The indefinite convolution integral exists

$$\int_{\overline{a}}^{\overline{b}} \overline{g}\, d\overline{u} = \overline{f}(\overline{u}).$$

The definite integral has meaning in function space but will not be discussed here. The transforms are corresponding integrals in Variational Calculus and are discussed in [36]. Convolution integrals are not used in this treatise.

10. COMPATIBILITY EQUATIONS

To introduce the concept, we consider again the analogy between a convolution function of a convolution variable and a complex function w of a complex variable $z = x + \hat{\imath}\, y$, which is also a function of the two variables x and y. The derivative $\frac{dw}{dz}$ must be the same for all directions of dz, particularly $\frac{dw}{dz} = \frac{dw}{dx} = \frac{dw}{d\hat{\imath}y}$, which leads to the Cauchy Riemann equations. We apply this approach now to a convolution function of a convolution variable. Consider any arbitrary functions $\{f_0, f_1, f_2 \ldots\}$ of the variables $\{u_0, u_1, u_2 \ldots\}$,

$$\overline{f}(\overline{u}) = \begin{bmatrix} f_0(\overline{u}) \\ f_1(\overline{u}) \\ f_2(\overline{u}) \\ \vdots \end{bmatrix} = \begin{bmatrix} f_0(u_0, u_1, u_2 \ldots) \\ f_1(u_0, u_1, u_2 \ldots) \\ f_2(u_0, u_1, u_2 \ldots) \\ \vdots \end{bmatrix}.$$

We want to investigate under which conditions the vector function $\overline{f}(\overline{u})$ is an analytic convolution function of the convolution variable $\overline{u}$. The dependent differential $d\overline{f}$ due to an arbitrary differential $d\overline{u}$, $d\overline{f} = \overline{f}(\overline{u} + d\overline{u}) - \overline{f}(\overline{u})$, can be expressed in matrix algebra as

$$d\overline{f} = \frac{d\overline{f}}{d\overline{u}} \cdot d\overline{u}, \tag{71}$$

where the vector derivative is the Jacobian matrix

$$\frac{d\overline{f}}{d\overline{u}} = \begin{bmatrix} \frac{\partial f_0}{du_0} & \frac{\partial f_0}{du_1} & \frac{\partial f_0}{du_2} & \cdots \\ \frac{\partial f_1}{du_0} & \frac{\partial f_1}{du_1} & \frac{\partial f_1}{du_2} & \cdots \\ \vdots & & & \ddots \end{bmatrix}.$$

The analytic convolution derivative, if it exists, is a convolution number

$$d\overline{f}/d\overline{u} = \overline{g} = \{g_0, g_1, g_2 \ldots\},$$

therefore, we require that

$$d\overline{f} = \overline{g} * d\overline{u}. \tag{72}$$

The convolution derivative is analytic if equation (72) is true for any realization of the differential $d\overline{u}$, or in terms of space, for any direction of the differential $d\overline{u}$, particularly for

$$d\overline{u} = \{0, 0, \ldots dx_i, 0, \ldots\} = \overline{\delta}_i dx_i \quad \text{(no sum intended)}.$$

For this direction, the matrix equation (71) becomes

$$d\overline{f} = \left.\frac{d\overline{f}}{d\overline{u}}\right|_{\text{column } i} \cdot dx_i,$$

and the convolution equation (72) becomes

$$d\overline{f} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ g_0 \\ g_1 \\ g_2 \\ \vdots \end{bmatrix} \cdot dx_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \overline{g} \\ \\ \\ \end{bmatrix} \cdot dx_i,$$

(with an arrow labelled i pointing to the row of g_0, resp. $\overline{g}$)

where the whole column $\overline{g}$ is shifted by the leading zeros of $\overline{\delta}_i$. This is true for every direction $du_i, i = 0, 1, \ldots$, therefore, the Jacobian matrix $\frac{d\overline{f}}{d\overline{u}}$ consists of identical columns $\overline{g}$ arranged as a semicirculant matrix

$$\frac{d\overline{f}}{d\overline{u}} = \begin{bmatrix} \overline{g} & & & & \\ & \overline{g} & & & \\ & & \overline{g} & & \\ & & & \overline{g} & \\ & & & & \ddots \end{bmatrix} = \begin{bmatrix} g_0 & & & & \\ g_1 & g_0 & & & \\ g_2 & g_1 & g_0 & & \\ g_3 & g_2 & g_3 & g_0 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{73}$$

This is a generalization of the semicirculant matrix which occurs when the linear function $\overline{a} * \overline{u}$ is written as a matrix equation [8]. We put the result in a theorem:

10.1.

THEOREM 2. *A vector function $\overline{f}(\overline{u})$ of the variable $\overline{u}$ is a convolution function of the convolution variable $\overline{u}$, and the derivative exists, if the Jacobian matrix $\dfrac{d\overline{f}}{d\overline{u}}$ is a semicirculant matrix, and the subcolumns consist of the convolution derivative $d\overline{f}/d\overline{u} = \overline{g}$.*

This theorem can be put in the form

$$\begin{aligned} \frac{\partial f_i}{du_j} &= 0, \qquad \text{for } j > i, \\ \frac{\partial f_{i+j}}{du_i} &= g_j, \qquad \text{for all } i \text{ and } j = 0, 1, 2, \ldots, \end{aligned} \tag{74}$$

which we call the *Compatibility Equations.* The theorem is true for regular and singular convolution numbers provided all operations are compatible. But for singular convolution numbers many of the terms are zero, and for such cases the theorem can be stated in a modified form to be useful.

Note that we don't need Theorem 2 or the functional forms $f_0(u_0, u_1, \ldots)$, $f_1(u_0, u_1, \ldots), \ldots$, for practical purposes. If a function $f(u(x), x)$ is given, we can immediately transform to the analytic convolution function by Theorem 1. We need Theorem 2 solely for theoretical basis of solution routines that we are going to develop to solve nonlinear convolution equations.

It is easy to see that all the arithmetic routines that have been given before satisfy the compatibility equations. Similarly, a sequence of arithmetic operations can always be written as a matrix product of a semicirculant matrix and the previous convolution number as column vector. With these observations then we are satisfied that all convolution operations as defined previously satisfy their own compatibility equations.

11. THEORY OF SOLUTION OF NONLINEAR CONVOLUTION EQUATIONS

Consider the nonlinear convolution equation

$$\overline{f}(\overline{u}) = \overline{0}, \tag{75}$$

where we take the third order polynomial $\overline{f}(\overline{u}) = \overline{a} + \overline{b} * \overline{u} + \overline{c} * \overline{u}^{*2} + \overline{d} * \overline{u}^{*3}$ as example. To develop the theory, we assume that $\overline{u}$ is a regular convolution number, $u_0 \neq 0$, and only later will the changes due to singular convolution numbers with any number of leading zeros be included, see Section 11.5.

The expansion of the first five terms is shown in Appendix 5, where the properties of Theorem 2 as set out by equation (74), can be verified. The elements f_i are algebraic functions of the elements $(u_0, u_1, \ldots)$, but in numerical work this algebraic expansion is completely avoided.

Consider three possible iteration methods to solve for the convolution number $\overline{u}$. Firstly the Newton Raphson iteration may be applied as if equation (75) was a general vector equation for which the iteration procedure is

$$\overline{u}_{i+1} = \overline{u}_i - \left[\frac{d\overline{f}}{d\overline{u}}\right] \backslash \overline{f}(\overline{u}_i),$$

where the [matrix]\ indicates left division by the matrix, implemented by Gauss elimination. This method works well with widely varying initial estimates, but is clumsy because it doesn't take advantage of the convolution properties. A large amount of work is required to express all

the derivatives $\frac{\partial f_i}{du_j}$ analytically and the translating them into program language. The matrix is triangular but a standard Gauss elimination routine doesn't recognize this because it is not upper triangular. At this stage, there is no indication of how to select initial values and how many different solutions may be found.

As a logical next method, we may consider using the Newton Raphson iteration using convolution algebra

$$\overline{u}_{i+1} = \overline{u}_i - \overline{f}(\overline{u}_i)/\left(d\overline{f}/d\overline{u}\right).$$

This method only needs the analytic derivative which is easily expressed in terms of convolution algebra for all known functions, the rest of the work can be programmed purely in terms of convolution number routines. This method is actually only a more efficient implementation of the previous method since the Jacobian matrix columns consist of the derivative $d\overline{f}/d\overline{u}$. The method works just as well. Still the same problem of initial values and number of different solutions remains.

None of the two methods above however takes advantage of the peculiarities of the convolution functions. From the expansion of the example function in Appendix 5 we make the following observation:
The function $f_0(\overline{u})$ and only this function contains only the variable u_0. In fact, the function $f_0(u_0)$ is exactly the same as the function in the number domain, $f(u)$, which becomes clear when we consider the function in the number domain,

$$\begin{aligned} f(u(x),x) &= f(u_0+u_1x+u_2x^2+\cdots,\ a_0+a_1x+a_2x^2+\cdots,\ b_0+b_1x+b_2x^2+\cdots,\ c_0+c_1x+c_2x^2+\cdots) \\ &= f_0 + f_1x + f_2x^2 \ldots \end{aligned}$$

for the particular value $x = 0$. This property is stated in the following theorem.

11.1.

THEOREM 3. *If $\overline{u}$ is the Taylor transform of the function $u(x)$, and $\overline{f}$ is the Taylor transform of the functional function $f(u(x),x)$, then the element f_0 is a function of the element u_0 only, and the function $f_0(u_0) = f(u(0),0)$.*

The same remarks as for Theorem 2 for convolution numbers with leading zero elements apply.

The function $f(u(0),0)$ is the simpler function $f(u(0),(a(0),b(0),c(0),\ldots))$. The root of this function is $u(0) = u_0$ which must be found by ordinary algebraic routines, generally iteration routines. Only the roots within the circle of convergence of f are valid. Particularly, for the case of a scalar polynomial of degree n, there are exactly n roots, some or all of which may be repeated. For a functional polynomial of degree n the conditions are slightly different. Using the illustration of Appendix 5, we see that the element u_1 is only contained linear in the element f_1 together with the now known u_0, so that we can solve for u_1 from the equation $f_1 = 0$, and there is only one solution of u_1 for each u_0; for a scalar polynomial equation this solution is 0. Next, we see that the element u_2 is only contained linear in the element f_2 together with the now known u_0 andu_1, so that we can solve for u_2 from the equation $f_2 = 0$, and there is only one solution of u_2 for each u_0; again for a scalar polynomial equation this solution is 0. This pattern continues for all higher elements. In fact, by comparing the factors of the higher elements u_i in each element of f_j and comparing the factors with the expansion of $\overline{g}$, we can arrange the terms in the elements of $\overline{f}$ as

$$\begin{aligned} f_0 &= a_0 + b_0u_0 + c_0u_0^2 + d_0u_0^3, \\ f_1 &= a_1 + b_1u_0 + c_1u_0^2 + d_1u_0^3 \\ &\quad +g_0.u_1, \end{aligned}$$

$$
\begin{aligned}
f_2 &= a_2 + b_2u_0 + c_2u_0^2 + d_2u_0^3 \\
&\quad + b_1u_1 + 2c_1u_0u_1 + c_0u_1^2 + 3d_1u_0^2u_1 + 3d_0u_0u_1^2 \\
&\quad + g_0.u_2, \\
f_3 &= a_3 + b_3u_0 + c_3u_0^2 + d_3u_0^3 \\
&\quad + b_2u_1 + 2c_2u_0u_1 + c_1u_1^2 + 3d_2u_0^2u_1 + 3d_1u_0u_1^2 + d_0u_1^3 \\
&\quad + g_1.u_2 + g_0.u_3, \\
f_4 &= a_4 + b_4u_0 + c_4u_0^2 + d_4u_0^3 \\
&\quad + b_3u_1 + 2c_3u_0u_1 + c_2u_1^2 + 3d_3u_0^2u_1 + 3d_2u_0u_1^2 + d_1u_1^3 \\
&\quad + b_2u_2 + 2c_2u_0u_2 + 2c_1u_1u_2 + c_0u_2^2 + 3d_2u_0^2u_2 + 6d_1u_0u_1u_2 + 3d_0u_1^2u_2 + 3d_0u_0u_2^2 \\
&\quad + g_1.u_3 + g_0.u_4, \\
f_5 &= a_5 + b_5u_0 + c_5u_0^2 + d_5u_0^3 \\
&\quad + b_4u_1 + 2c_4u_0u_1 + c_3u_1^2 + 3d_4u_0^2u_1 + 3d_3u_0u_1^2 + d_2u_1^3 \\
&\quad + b_3u_2 + 2c_3u_0u_2 + 2c_2u_1u_2 + c_1u_2^2 + 3d_3u_0^2u_2 + 6d_2u_0u_1u_2 + 3d_1u_1^2u_2 + 3d_1u_0u_2^2 + 3d_0u_1u_2^2 \\
&\quad + g_2u_3 + g_1.u_4 + g_0.u_5, \\
&\vdots
\end{aligned}
$$

We use the theory again to show what the general pattern will be. The compatibility equations for the derivative $\overline{h} = d\overline{g}/d\overline{u}$ of the derivative $\overline{g}\,(\overline{u})$ are

$$
\frac{d\overline{f}}{d\overline{u}} = \begin{bmatrix} \frac{\partial g_0}{du_0} & \frac{\partial g_0}{du_1} & \frac{\partial g_0}{du_2} & \cdots \\ \frac{\partial g_1}{du_0} & \frac{\partial g_1}{du_1} & \frac{\partial g_0}{du_2} & \cdots \\ \frac{\partial g_2}{du_0} & \frac{\partial g_0}{du_1} & \frac{\partial g_2}{du_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} h_0 & & & \\ h_1 & h_0 & & \\ h_2 & h_1 & h_0 & \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{76}
$$

which we use to make the statements:

$\frac{\partial f_1}{du_0} = g_1$, $\frac{\partial f_1}{du_1} = g_0$, $\frac{\partial f_1}{du_{i>1}} = 0$, therefore, $f_1(\overline{u})$ contains u_0, u_1 only.

$\frac{\partial g_0}{du_1} = 0$, therefore, $f_1(u_0, u_1)$ contains u_1 only linear and can be written $f_1 = f_1(u_0, 0) + g_0u_1$.

$\frac{\partial f_2}{du_0} = g_2$, $\frac{\partial f_2}{du_1} = g_1$, $\frac{\partial f_2}{du_2} = g_0$, $\frac{\partial f_2}{du_{i>2}} = 0$, therefore, $f_2(\overline{u})$ contains u_0, u_1, u_2 only.

$\frac{\partial g_0}{du_2} = 0$, therefore, $f_2(u_0, u_1, u_2)$ contains u_2 only linear and can be written

$f_2 = f_2(u_0, u_1, 0) + g_0u_2$.

$\frac{\partial f_3}{du_0} = g_3$, $\frac{\partial f_3}{du_1} = g_2$, $\frac{\partial f_3}{du_2} = g_1$, $\frac{\partial f_3}{du_3} = g_0$, $\frac{\partial f_3}{du_{i>3}} = 0$, therefore, $f_3(\overline{u})$ contains u_0, u_1, u_2, u_3 only.

$\frac{\partial g_1}{du_2} = 0$, $\frac{\partial g_0}{du_3} = 0$, therefore $f_3(u_0, u_1, u_2, u_3)$ contains u_3, u_2 only linear and can be written

$f_3 = f_3(u_0, u_1, 0, 0) + g_1u_2 + g_0u_3$.

These results are summarized

$$
\begin{aligned}
f_0 &= f_0(u_0) \\
f_1 &= f_1(u_0, 0) + g_0u_1
\end{aligned}
$$

$$
\begin{aligned}
f_2 &= f_2(u_0, u_1, 0) + g_0 u_2 \\
f_3 &= f_3(u_0, u_1, 0, 0) + g_1 u_2 + g_0 u_3 \\
f_4 &= f_4(u_0, u_1, u_2, 0, 0) + g_1 u_3 + g_0 u_4 \\
f_5 &= f_5(u_0, u_1, u_2, 0, 0, 0) + g_2 u_3 + g_1 u_4 + g_0 u_5 \\
f_6 &= f_6(u_0, u_1, u_2, u_3, 0, 0, 0) + g_2 u_4 + g_1 u_5 + g_0 u_6 \\
&\vdots
\end{aligned}
\tag{77}
$$

We put these results in a theorem which is a extension of Theorem 3.

11.2.

THEOREM 4. *If $\overline{u}$ is the Taylor transform of the function $u(x)$, and $\overline{f}$ is the Taylor transform of the functional function $f(u(x), x)$, then the elements f_i are functions of the elements $u_{j \le i}$ only, and the highest k elements u_j occur linear only, where $k = \text{int}\left(\dfrac{i-1}{2}\right)$, with the lower elements $u_{j<i-k}$ as coefficients.*

Particularly, it means that if a Taylor expansion of a function of $u(x)$ is required up to the n^{th} term, then in none of the algebraic operations is any expansion needed to higher than the n^{th} term, provided the convolution number $\overline{u}$ is regular. The modification for singular convolution numbers will be given in Section 14.

This result is at least in part well-known and applied in many practical Taylor expansions, also where the elements are functions, discussed in Section 22. Note that this only applies to algebraic functional functions, i.e., where no integration and differentiation w.r.t. the number domain variable x is involved, and the same remarks as for Theorems 2 and 3 for convolution numbers with leading zero elements apply.

Equation (77) shows that the theoretical solution procedure for any nonlinear convolution equation $\overline{f}(\overline{u}) = \overline{0}$ can be written in the form

$$
\begin{aligned}
u_0 &= f_0^{[-1]}(0) \\
u_1 &= -f_1(u_0, 0)/g_0 \\
u_2 &= -f_2(u_0, u_1, 0)/g_0 \\
u_3 &= -f_3(u_0, u_1, u_2, 0)/g_0 \qquad \text{if } g_0 \neq 0, \\
u_4 &= -f_4(u_0, u_1, u_2, u_3, 0)/g_0 \\
u_5 &= -f_5(u_0, u_1, u_2, u_3, u_4, 0)/g_0 \\
&\vdots
\end{aligned}
\tag{78}
$$

where $f_0^{[-1]}$ indicates the reverse function, see Section 20.1. We call the element g_0 a pivot element. It plays the role of a pivot, as in the division and square root routines, similar to the pivot element in a matrix division. The solution method is recursive as the solution of each element requires the previous elements to be known. In the later sections, we describe how to solve the equations numerically without the analytical burden of having to the determine the algebraic forms of the sequence of functions f_i, which is the purpose of the concept of convolution number compared to symbolic computation with polynomials.

The advantage of a solution method based on these equations to the previous two is obvious: only one scalar nonlinear equation needs to be solved nonlinear, and the conclusion that number of different solutions are equal to the number of different leading element solutions.

In the solution of nonlinear equations, the leading element is usually computed iteratively, and therefore requires an infinite number of steps to obtain the exact value. The strict definition of

the solution as convolution number is therefore lost, e.g., that the elements are exact because they are determined in a finite number of steps. This also creates a problem if the numerical routines are implemented symbolically to obtain symbolic analytic expressions for the elements. We can only give the leading element a symbolic name and then derive the remaining elements symbolically in terms of this unmentionable element.

Nevertheless, we still obtain a machine exact result. By the same token, a computer doesn't distinguish between rational and irrational numbers.

For the polynomial equation $\overline{f}(\overline{u}) = \overline{0}$ of degree n, we can make some definite statements: If the polynomial is scalar then the convolution number consists only of the n, possible solutions of the element u_0. If the polynomial has functional coefficients, then different cases can occur.

If the roots u_0 of the leading equation $f_0(u_0) = 0$ are distinct, then there are as many convolution solutions $\overline{u}$ as there are roots. These may be less than the degree of the convolution polynomial, e.g., in Example 2 of Section 5.1 the polynomial is of second degree, the explicit solution involved the square root which allows two solutions, but it was found that only one was possible—as was to be expected from the posed physical problem. The number of roots depend on the highest power for which the leading element of the coefficient is not zero, i.e., the degree of the polynomial $f(u_0, x)$ when $x = 0$.

Consider now the case where $g_0 = 0, g_1 \neq 0$. Since g_0 is a function of u_0 only, this happens when $\dfrac{\partial f_0}{du_0} = 0$ at u_0, which means that u_0 is a repeated root. The whole sequence of equations (77) become

$$\begin{aligned}
f_0 &= f_0(u_0) \\
f_1 &= f_1(u_0, 0) \\
f_2 &= f_2(u_0, u_1, 0) \\
f_3 &= f_3(u_0, u_1, 0, 0) + g_1 u_2 \\
f_4 &= f_4(u_0, u_1, u_2, 0, 0) + g_1 u_3 \\
f_5 &= f_5(u_0, u_1, u_2, 0, 0, 0) + g_2 u_3 + g_1 u_4 \\
f_6 &= f_6(u_0, u_1, u_2, u_3, 0, 0, 0) + g_2 u_4 + g_1 u_5 \\
&\vdots
\end{aligned} \tag{79}$$

and the solution routine is

$$\begin{aligned}
& u_0 = f_0^{[-1]}(0) \\
\text{if } & f_1(u_0, 0) = 0 \quad \text{then:} \\
& u_1 = f_2^{[-1]}(0) \\
& u_2 = -f_3(u_0, u_1, 0, 0)/g_1 \quad \text{if } g_0 = 0 \\
& u_3 = -f_4(u_0, u_1, u_2, 0, 0)/g_1 \quad \text{and } g_1 \neq 0 \\
& u_4 = -f_5(u_0, u_1, u_2, u_3, 0, 0)/g_1 \\
& u_5 = -f_6(u_0, u_1, u_2, u_3, u_4, 0, 0)/g_1 \\
& \vdots
\end{aligned} \tag{80}$$

The second in the set of equations (80) must be compatible otherwise no convolution solution $\overline{u}$ exists with this leading element u_0. This does not mean that no solution $u(x)$ exists; it means that no Taylor expansion of $u(x)$ exists. This condition places restrictions on the coefficient functions. As can be seen from the example, $f_1(u_0, 0)$ contains the element 1 of their expansions. The element u_1 is now determined from f_2 which is a quadratic equation in u_1 and therefore has two solutions. The remaining elements are determined by the remaining linear equations in the

unknowns u_i from equations (81) with now g_1 as the pivot element. The result are two generally different solutions $\overline{u}$ for the double root u_0. An example of a repeated root u_0 in f_0 and not compatible f_1 is the equation $(1+x)+(2+x)u(x)+(1+x)u^2(x) = 0$. The repeated root $u_0 = -1$ but $f_1(u_0) = 1 \neq 0$ therefore no solution $\overline{u}$ exists. The function $u(x)$ can be determined explicitly and is found to contain $\sqrt{x}$ which prevents the Taylor expansion of any of the two solutions. The same pattern continues if the next pivot element g_1 is also zero. This may happen if u_0 is a triple root of $f_0(u_0)$ which means that $\dfrac{\partial f_0}{du_0} = g_0 = 0$ and $\dfrac{\partial^2 f_0}{du_0^2} = \dfrac{\partial g_0}{du_0} = h_0 = 0$.

Now $\dfrac{\partial f_2}{du_1} = g_1 = 0$, i.e., the coefficient of u_1 in f_2 is zero. By a similar analysis the coefficient u_2 does not occur in g_2 and linear in f_4. For $g_0 = 0$, $g_1 = 0$, $g_2 \neq 0$, the sequence of equations (77) become, written already in the form of solution sequence,

$$
\begin{aligned}
u_0 &= f_0^{[-1]}(0) \\
\text{if } f_1(u_0,0) &= 0, \qquad \text{then:} \\
\text{if } f_2(u_0,0,0) &= 0, \qquad \text{then:} \\
u_1 &= f_3^{[-1]}(0) \quad \text{if } g_0 = 0 \\
u_2 &= -f_4(u_0,u_1,0,0,0)/g_2 \quad \text{and } g_1 = 0 \\
u_3 &= -f_5(u_0,u_1,u_2,0,0,0)/g_2 \quad \text{and } g_2 \neq 0 \\
u_4 &= -f_6(u_0,u_1,u_2,u_3,0,0,0)/g_2 \\
&\vdots
\end{aligned}
\tag{81}
$$

If $g_1 = 0$ then $\dfrac{\partial f_1}{du_0} = 0$ and $\dfrac{\partial f_2}{du_1} = 0$, therefore, u_0 is a double root of $f_1(u_0,0)$ so that $f_1 = 0$. Now $f_2(u_0)$ must be compatible, i.e., u_0 must be a root of f_2, otherwise no solution exists. The element u_1 is determined from f_2 which is a cubic equation in u_1 and, therefore, has three solutions. The remaining elements are determined by the remaining linear equations in the unknowns u_i arranged like in equations (78) with g_2 as the pivot element. Again g_2 doesn't contain u_2 because now $\dfrac{\partial g_2}{du_2} = h_0 = 0$ therefore the first step of division by g_2 is possible. The result are three generally different solutions $\overline{u}$ for the triple root u_0.

The highest power to which element u_i may occur in the function f_k can be determined from the compatibility equations as follows, writing $\overline{g}$, $\overline{h}$, $\overline{l}$, ... for the consecutive convolution derivatives of $\overline{f}$.

$$
\begin{aligned}
\frac{\partial f_k}{du_i} &= 0, \quad \text{for } k < i, \ f_k \text{ doesn't contain } u_i \\
&= g_{k-i} \neq 0 \quad \text{for } k \geq i, \ f_k \text{ contains } u_i, \\
\frac{\partial g_{k-i}}{du_i} &= 0, \quad \text{for } k < 2i, \ f_k \text{ contains } u_i^1 \quad \text{for } i \leq k < 2i \\
&= h_{k-2i} \neq 0 \quad \text{for } k \geq 2i, \ f_k \text{ contains } u_i^2, \\
\frac{\partial h_{k-2i}}{du_i} &= 0, \quad \text{for } k < 3i, \ f_k \text{ contains } u_i^2 \quad \text{for } 2i \leq k < 3i \\
&= l_{k-3i} \neq 0 \quad \text{for } k \geq 3i, \ f_k \text{ contains } u_i^3, \\
\frac{\partial l_{k-3i}}{du_i} &= 0, \quad \text{for } k < 4i, \ f_k \text{ contains } u_i^3 \quad \text{for } 3i \leq k < 3i \\
&\vdots
\end{aligned}
$$

Therefore,

$$\begin{array}{c} \text{for } i \leq k < (n+1)\, i \\ f_k \text{ contains } u_i^n \quad \text{if the polynomial has corresponding coefficients} \end{array} \tag{82}$$

These results can easily be used to show that the functions $f_i(\overline{u})$ are multivariate polynomials in the variables u_i which is shown in a different way in [42]. In our numerical treatment, we never use this fact though.

In any case, there are always as many solutions for u_1 possible as the multiplicity of the roots u_0 but not more, and the remaining higher elements occur linear again. Without covering the possibility when there are multiple roots of u_1 again, we indicate the situation of those cases. For any multiplicity of u_1, the next equation will not contain u_2 and must be compatible. The next equation that does contain u_2 will appear to the degree of the multiplicity of the particular root u_1. If distinct solutions of u_2 exist, all higher elements occur linear again. If multiple roots of u_2 exist, then we embark on a new branch of an ever expanding tree structure of possibilities, which we don't explore any further here. However, from the previous pattern we can see that we never obtain more repeated roots of the new element than the multiplicity of the previous root until at most as many different branches exist as the multiplicity of the root u_0 of f_0, after which the remaining elements are determined linearly from the previous roots for each branch. The extreme case occurs when all elements are obtained from nonlinear equations until we reach the truncation element. In that case, $\overline{u}$ is a multiple convolution root of the convolution function because then $\overline{g} = \overline{0}$, at least up to the truncated length, similar to a multiple root of a number equation which can only be estimated within machine accuracy of the truncated digital number used for computing. As an example, we may reconstruct the steps that will be necessary to solve the convolution equation $(\overline{u} - \overline{a})^{*2} = \overline{0}$ by the described sequence of steps. We find that every second equation is the same quadratic equation for a new element with only one repeated root, and every other second equation is compatible. The recursive solution method is based on the first convolution derivative $\overline{g}$, and comparing with root solving of a nonlinear function in the number domain we notice the similarity that a method using the first derivative fails at multiple roots.

Because of all the special cases where no solution exists the Fundamental Theorem of Algebra does not apply to convolution polynomials. Without attempting a general proof that includes all possible cases, we rely on the proved cases to state the following theorem.

11.3.

THEOREM 5. *A convolution polynomial $\overline{p}(\overline{u})$ of degree $m \geq n$ has n distinct convolution roots if the n roots u_0 of the leading scalar polynomial $p_0(u_0)$ of degree n are distinct. If some roots u_0 are of multiplicity r then there may be r distinct or multiple convolution roots for each u_0, or less.*

The theorem can be applied to any function that is not a polynomial if the n roots of the function within the circle of convergence are used.

Before a suggested solution scheme is introduced, we have to address the problem of finding in the first place the Taylor series of functions that occur as coefficients in functional equations, i.e., whose Taylor transforms occur as convolution coefficients in convolution equations. We certainly don't look them up in tables—the purpose of this treatise is to produce them, directly in numerical values. But before that differential equations must be discussed.

11.4. Several Variables

Applying Theorem 1 to a function of several variables $f(u(x), v(x), \ldots)$, we obtain the Taylor transform $\overline{f}(\overline{u}, \overline{v}, \ldots)$, which is a convolution function of several convolution variables,

where it doesn't matter whether any of the quantities u, v, ... are regarded as variables or constants.

Similar Theorem 2 applies to each of the variables, the Jacobian matrices now being

$$\frac{\partial \overline{f}}{d\overline{u}}, \frac{\partial \overline{f}}{d\overline{v}}, \ldots .$$

In Theorem 3, the statement $f_0(u_0) = f(u(0), 0)$ becomes simply $f_0(u_0, v_0, \ldots) = f(u(0), v(0), \ldots, 0)$ by the same argument that leads to Theorem 3. Let now the convolution derivatives

$$\partial \overline{f} \,/\, d\overline{u} = \overline{g}, \quad \partial \overline{f} \,/\, d\overline{v} = \overline{r}, \ldots,$$

then the derivatives

$$\frac{\partial f_0}{du_0} = g_0, \quad \frac{\partial f_0}{dv_0} = r_0, \ldots,$$

while

$$\frac{\partial f_{i>0}}{du_0} = 0, \quad \frac{\partial f_{i>0}}{dv_0} = 0, \ldots,$$

which confirms Theorem 3 for several variables.

Theorem 4 remains the same w.r.t. $\overline{u}$ and $\overline{v}$, but now we want to confirm that the coefficient of the highest element u_i in f_i doesn't include the element v_i or any higher element of $\overline{v}$, as is evident from the sample functions in Appendix 5 if we consider any of the constants there as variables.

From $\dfrac{\partial f_i}{du_i} = \dfrac{\partial f_0}{du_0} = g_0$, it is clear that in f_i only v_0 can occur as coefficient of u_i. Vice versa, from $\dfrac{\partial f_i}{dv_i} = \dfrac{\partial f_0}{dv_0} = r_0$, it is clear that in f_i only u_0 can occur as coefficient of v_i. Therefore, Theorem 3 for several variables means that in the element f_i, the elements $u_i, v_i, \ldots$ occur linear only. For the solution of nonlinear convolution equations

$$\begin{aligned} \overline{f}\left(\overline{u}, \overline{v}, \ldots\right) &= 0, \\ \overline{p}\left(\overline{u}, \overline{v}, \ldots\right) &= 0 \\ &\vdots \end{aligned}$$

it means that the leading elements $u_0, v_0 \ldots$ are solved from the same number of nonlinear equations

$$\begin{aligned} f_0(u_0, v_0, \ldots) &= 0, \\ p_0(u_0, v_0, \ldots) &= 0 \\ &\vdots \end{aligned}$$

and then the elements $u_i, v_i, \ldots$ are solved recursively from the same number of linear equations

$$\begin{aligned} f_i(u_i, v_i, \ldots) &= 0, \\ p_i(u_i, v_i, \ldots) &= 0 \\ &\vdots \end{aligned}$$

12. TRANSFORMATION OF DERIVATIVES

Let a function be $u(x)$ and its Taylor transform $\overline{u}$. The derivative in the number domain transforms to the well-known particular simple operation in the convolution domain

$$v \equiv \frac{du}{dx} \equiv u'(x) = \{u_1, 2u_2, 3u_3, \ldots\} \equiv \{v_0, v_1, v_2, \ldots\}. \tag{83}$$

This operation cannot be expressed as a convolution function, i.e., $\overline{v}$ is not a convolution function of $\overline{u}$, as can easily be ascertained by checking the compatibility equations in the Jacobian matrix $\frac{\partial \overline{v}}{d\overline{u}}$. This must be so since the derivative is not an algebraic operation on function values.

We use the same derivative symbol $'$ for the Taylor transform of a derivative, i.e.,

$$u'(x) = \underline{X} \cdot \overline{u}',$$
$$u''(x) = \underline{X} \cdot \overline{u}'',$$

where on the left the symbol $'$ stands for the number domain derivative and on the right for the corresponding transformed operation on the convolution number. Because of previous definitions, we may not call it derivative on both sides. We have a conflict of terminology inasmuch as addition and multiplication can have the same name in both domains by simply defining the convolution a multiplication; yet the convolution derivative was defined by an isomorphism, not by the transform, which turned out to be the variational. To overcome this problem, we resort to a rather primitive name and call the transformed derivative *prime*, from the prime symbol as notation for the derivative (compare "dot product"). We can then at least verbally use the term prime in both domains: the Taylor transform of "$u(x)$-prime is $\overline{u}$-prime."

The integral in the number domain transforms to the reverse simple operation in the convolution domain, and as reverse of the prime symbol we use a ^{I} symbol,

$$w \equiv \int u(x)\,dx = u^I(x) = \left\{w_0, u_0, \tfrac{1}{2}u_1, \tfrac{1}{3}u_2, \ldots\right\}, \equiv \{w_0, w_1, w_2, w_3, \ldots\}. \tag{84}$$

We use the same new integral symbol ^{I} for the Taylor transform of an integral, i.e.,

$$u^I(x) = \underline{X} \cdot \overline{u}^I$$
$$u^{II}(x) = \underline{X} \cdot \overline{u}^{II}$$

Explicitly, the formulas for transforms of derivatives and integrals are

$$u_i' = (i+1)\,u_{i+1} \qquad \text{for } i = 0, 1, \ldots, n-1, \tag{85}$$
$$u_i^I = \frac{u_{i-1}}{i} \qquad \text{for } i = 1, \ldots, n, \quad \text{value of } u_0 \text{ supplied.} \tag{86}$$

The value u_0 should be a necessary argument in the integration routine. This will remind the programmer to supply the initial value, and it makes an additional statement in another line unnecessary. When these two routines are programmed, the set of basic routines for convolution analysis is complete.

Yet these symbols are not enough, we must still distinguish between a total derivative and the partial derivative as used in the chain-sense for functional functions. For this we use the subscript x and u in both domains, a convention that is quite unsatisfactory because it is dependent

on the particular variables used in a problem but which has been firmly entrenched in partial derivatives. Because it is sometimes useful to distinguish between result and operation in an equation, we even indicate the transformed operation on the convolution number with the same notation of the number domain and write

$$\overline{u}' = \frac{d}{dx}\overline{u},$$
$$\overline{u}^I = \int \overline{u}\, dx,$$

where the variables u and x must be defined beforehand in the context. The complete number domain and convolution domain notations for derivatives are then

$$\begin{array}{lrcl} \text{number domain:} & \dfrac{df}{dx} & = & \dfrac{\partial f}{\partial x} + \dfrac{\partial f}{\partial u}\dfrac{du}{dx}, \\ & f' & = & f_x + f_u u', \\ \text{convolution domain:} & d/dx\,\overline{f} & = & \overline{f}_x + d\overline{f}/d\overline{u} * \overline{u}', \\ & \overline{f}' & = & \overline{f}_x + \overline{f}_u * \overline{u}'. \end{array}$$

Note how the partial derivative of functions, in the function domain, is used for the variational.

We are now in a position to transform a differential or integro[10] equation of a function $u(x)$, linear with constant or functional coefficients, or nonlinear, e.g.,

$$\int u dx + u'' + a(x)u(u')^2 + f(u(x)) + g\,(u(x), x),$$

transforms to

$$\overline{u}^I + \overline{u}'' + \overline{a} * \overline{u} * \overline{u}'^{*2} + f(\overline{u}) + \overline{g}\,(\overline{u}).$$

Therefore, Theorem 1 is extended to the most general form that we need in this treatise as follows.

12.1.

THEOREM 6. *If the Taylor transform of a function $u(x)$ is*

$$u = \underline{X} \cdot \overline{u},$$

then a scalar, functional, differential and integro nonlinear equation $L(u(x))$ of a function $u(x)$ is transformed by

$$L(u) = \underline{X} \cdot \overline{L}(\overline{u}),$$

where $\overline{L}(\overline{u})$ is the same convolution operation on the convolution number $\overline{u}$ as $L(u)$ of the function $u(x)$ with the functional, differential and integral operations and variable coefficients on a scalar variable replaced by the transformed operations and convolution coefficients of a convolution variable.

With the present operations available, it is a simple matter to evaluate integrals, e.g., an elliptic integral, [5], in the notation used before,

$$\lambda = \int \frac{r_1 + r_2\sqrt{p}}{r_3 + r_4\sqrt{p}}\, dx$$

transforms to

$$\overline{\lambda} = \int \frac{\overline{r}_1 + \overline{r}_2\sqrt{\overline{p}}}{\overline{r}_3 + \overline{r}_4\sqrt{\overline{p}}}\, dx,$$

[10]Meaning containing indefinite integrals, but not integral equations.

where all convolution operations are performed by the prescribed routines, and $\lambda(0)$ entered as initial value in the integration step. There is not even a saving of computation by reducing the integral to parts as in the theory of elliptic integrals in the number domain. Also, the polynomial may be higher than fourth degree so that reduction to standard elliptic integrals may not be possible. The roots of $p(x)$ must be found in a separate rootfinding routine to establish the radius of convergence. The best length of the convolution number and the useful region must still be found by experimenting. Typical values are $n = 25$ and 0.7 of the radius of convergence for seven digit computation. Excellent results were obtained with $n = 50$ and 0.7 of the radius of convergence for 13 digit computation on the HP 9836 desk computer. This range covers many of the elliptic integrals that are required in applications, e.g., all the values that are tabulated in [43] for ring vortex calculations of potential flow aerodynamics. Should we need the functions for the complete range then the convolution numbers can be combined with the method of Frobenius.

13. THEORY OF SOLUTION OF DIFFERENTIAL EQUATIONS

We consider ordinary differential equations $L(u(x))$ of a univariate function $u(x)$ with initial conditions. It is well-known that Taylor expansions of solutions of differential equations can be obtained by recursive formula starting with the initial conditions. The analytic formal solution is given in [42]. We prove the recursive method here generally by using the compatibility equations of the transformed convolution equations.

Let $\overline{u}' = \overline{v}, \overline{u}'' = \overline{w}, \ldots$ then $\overline{u}, \overline{v}, \overline{w}$ are convolution variables and any convolution function $\overline{f}(\overline{u}), \overline{f}(\overline{v}), \overline{f}(\overline{w})$ must obey compatibility equations which are valid for algebraic convolution functions but not for the prime operation included in $\overline{L}(\overline{u})$. Write the prime operation on a convolution number in matrix form

$$\begin{aligned}\overline{u}' &= \begin{bmatrix} 0 & 1 & & & \\ & & 2 & & \\ & & & 3 & \\ & & & & \ddots \end{bmatrix} \cdot \overline{u} \\ &= \underline{\overline{N}} \cdot \overline{u},\end{aligned}$$

then we can write the differential operator in the form

$$\begin{aligned}\overline{L}(\overline{u}) &= \overline{f}\left(\overline{u}, \overline{v}, \overline{w}, \ldots\right) \\ &= \overline{f}\left(\overline{u},\ \underline{\overline{N}} \cdot \overline{v},\ \underline{\overline{N}}^2 \cdot \overline{w}, \ldots\right),\end{aligned}$$

and the vector derivative in terms of Jacobian matrices becomes

$$\begin{aligned}\frac{d\overline{f}}{d\overline{u}} &= \frac{\partial \overline{f}}{d\overline{u}} + \frac{\partial \overline{f}}{d\overline{v}} \cdot \frac{d\overline{v}}{d\overline{u}} + \frac{\partial \overline{f}}{d\overline{w}} \cdot \frac{d\overline{w}}{d\overline{u}} \cdots \\ &= \frac{\partial \overline{f}}{d\overline{u}} + \frac{\partial \overline{f}}{d\overline{v}} \cdot \underline{\overline{N}} + \frac{\partial \overline{f}}{d\overline{w}} \cdot \underline{\overline{N}}^2 \cdots \\ &= \begin{bmatrix} \overline{g} & & & \\ & \overline{g} & & \\ & & \overline{g} & \\ & & & \ddots \end{bmatrix} + \begin{bmatrix} \overline{r} & & & \\ & \overline{r} & & \\ & & \overline{r} & \\ & & & \ddots \end{bmatrix} \cdot \underline{\overline{N}} + \begin{bmatrix} \overline{s} & & & \\ & \overline{s} & & \\ & & \overline{s} & \\ & & & \ddots \end{bmatrix} \cdot \underline{\overline{N}}^2 \cdots\end{aligned}$$

$$= \left[\begin{array}{c|c|c|c|c|c} \overline{g} & \overline{r} & 2\overline{s} & & & \\ & \overline{g} & 2\overline{r} & 6\overline{s} & & \\ & & \overline{g} & 3\overline{r} & 12\overline{s} & \\ & & & \overline{g} & 4\overline{r} & \ddots \end{array}\right] + \cdots \tag{87}$$

where $\overline{g} = \overline{f}_u$, $\overline{r} = \overline{f}_v$, $\overline{s} = \overline{f}_w, \ldots$. The matrix is of an almost triangular structure. The first row is occupied with as many elements, counting from zero, as the order m of the differential equation, i.e., the order of the highest derivative, from the highest power of $\overline{N}$ that occurred, independent of the functional form in which the derivatives occurred. If the initial conditions $u(0), u'(0), u''(0), u'''(0), \ldots$, of the function $u(x)$ are known, the n leading elements $u_0, u_1, u_2, u_3, \ldots$, are known. Consider for a moment the remaining unknown elements u_m, u_{m+1}, $u_{m+2}, \ldots$, as a vector $\overline{u}_m$, then the Jacobian matrix $\dfrac{d\overline{f}}{d\overline{u}_m}$ with respect to the unknowns is the purely lower triangular matrix partition of the matrix $\dfrac{d\overline{f}}{d\overline{u}}$ above, explicitly

$$\frac{d\overline{f}}{d\overline{u}_m} = \begin{bmatrix} \dfrac{\partial f_0}{du_{m0}} & & & \\ \dfrac{\partial f_1}{du_{m0}} & \dfrac{\partial f_1}{du_{m1}} & & \\ \dfrac{\partial f_2}{du_{m0}} & \dfrac{\partial f_2}{du_{m1}} & \dfrac{\partial f_2}{du_{m2}} & \\ \vdots & & & \ddots \end{bmatrix},$$

which is similar to the matrix of compatibility conditions of a convolution function. By applying the same analysis of Section 11.2 for the solution of nonlinear convolution equations $\overline{f}(\overline{u})$ of the convolution number $\overline{u}$ now to the vector function $\overline{f}(\overline{u}_m)$ of the vector $\overline{u}_m$, we find some of the same conclusions:

f_k does not contain elements $u_{mi>mk}$.

f_k contains element u_{mk} only linear if $k > 0$.

f_0 is a nonlinear function of the lowest element u_m, to the same degree of nonlinearity as the element $u^{(m)}$ occurs in L.

The last nonlinearity falls away if the highest derivative occurs only linear in the differential equation, i.e., in an explicit differential equation. The conclusion is therefore the same for the Taylor expansion of the solution of differential equations with initial conditions as it is for nonlinear algebraic equations: when the first element u_m has been found, the remaining elements follow linear from a recursive method of solution. This coincides with the uniqueness of a solution of a differential equation. Surprisingly, then the functional nonlinear differential equations are not any more difficult to solve than linear differential equations with constant coefficients, keeping in mind that we only consider numerical solutions of the Taylor coefficients, the recurrence scheme just contains more algebraic operations. This property has long been an advantage in Taylor expansions, [44], and is discussed in what is termed the composition method in [18,22].

The actual method of solution can be carried out using different arrangements of computer storage. Special computer storage of the convolution number for this purpose is described in the next sections.

Obviously, we only obtain solutions which have Taylor expansions because we don't even include a finite number of leading negative powers as explained in Section 2. Also, we don't include the method of Frobenius for linear differential equations [29], but it can be combined with the method of convolution numbers.

14. POINTER CONVOLUTION NUMBER

The Pointer convolution number is the computational storage form of the convolution number with an additional pointer p that indicates up to which element a convolution number has been determined in a recursive solution scheme. The number is stored in an array of length $n + 3$ (counted from zero) as in Figure 6, where the three pointer convolution numbers $\overline{a}$, $\overline{b}$, $\overline{c}$ are shown.

The last three storage positions contain the pointers l, m, as before, and p. The elements of a convolution number $\overline{c}$ that have been determined as the result of a convolution operation of $\overline{a}$ and $\overline{b}$ may be known up to a position p_c. During a recursive loop the numbers $\overline{a}$ and $\overline{b}$ may have been determined up to positions p_a and p_b, both $> p_c$. This is the situation shown in Figure 6 "before." As was shown in the previous sections, the elements of $\overline{c}$ can now be determined up to an element which is not higher than the lesser of the pointers p_a and p_b, but modified by the shift that occurs due to leading zeros, which can be traced for each of the defined operations. In the following, we often call the pointer convolution numbers simply pointer numbers.

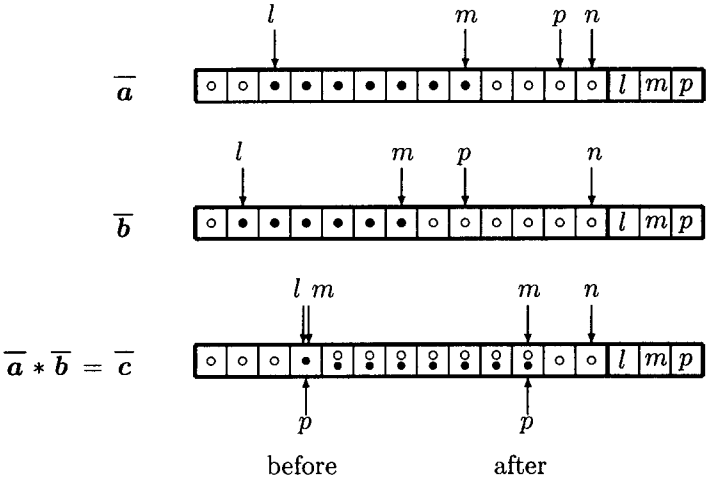


Figure 6. Pointer convolution number multiplication $\overline{a} * \overline{b} = \overline{c}$.

All the arithmetic, differentiation and integration routines are programmed for the pointer convolution numbers to start at element c_{p+1} instead of c_i where the leading element i is described in the routines. This change of program needs no more than reading the pointer values p_a, p_b and p_c at the beginning of the routine and changing the lower limit i to $p_c + 1$ and the upper limit to the new determined value of p_c. The pointers l and m are still present and, particularly, the pointers l are necessary to determine the corresponding pointer position p_c in each routine as shown later. At the end of the routine, the new pointer for the resulting pointer convolution number $\overline{c}$ is set to the new p_c.

In the pointer addition/subtraction routine, the new pointer in $\overline{c}$ is simply determined by the lesser of the pointers in $\overline{a}$ and $\overline{c}$,

$$p_c = \min(p_a, p_c). \tag{88}$$

The new limit of m in $\overline{c}$ must be reset and may never, under any circumstances, be higher than the new p_c. A typical pointer multiplication is shown in Figure 7, where the numbers on the diagonals indicated by a • are the new numbers that are computed to determine the new elements of $\overline{c}$ from the old pointer p_c onwards. Note how all elements of $\overline{a}$ and $\overline{b}$ are required for the new elements of $\overline{c}$.

The pointer positions in the multiplication routine can be traced from the diagram in Figure 7. The element p_a influences $\overline{c}$ in position $l_b + p_a$ and the element p_b influences $\overline{c}$ in position $l_a + p_b$. Therefore, the new pointer in $\overline{c}$ is determined as

$$p_c = \min(l_b + p_a,\ p_b + l_a). \tag{89}$$

Similarly, also from Figure 7, in the division routine $\overline{a} = \overline{c} / \overline{b}$, the pointer in $\overline{a}$ is determined by

$$p_a = \min(p_c - l_b,\ p_b + l_a), \tag{90}$$

which, for the square root routine $\overline{a} = \sqrt{\overline{c}}$, becomes

$$p_a = p_c - l_c/2. \tag{91}$$

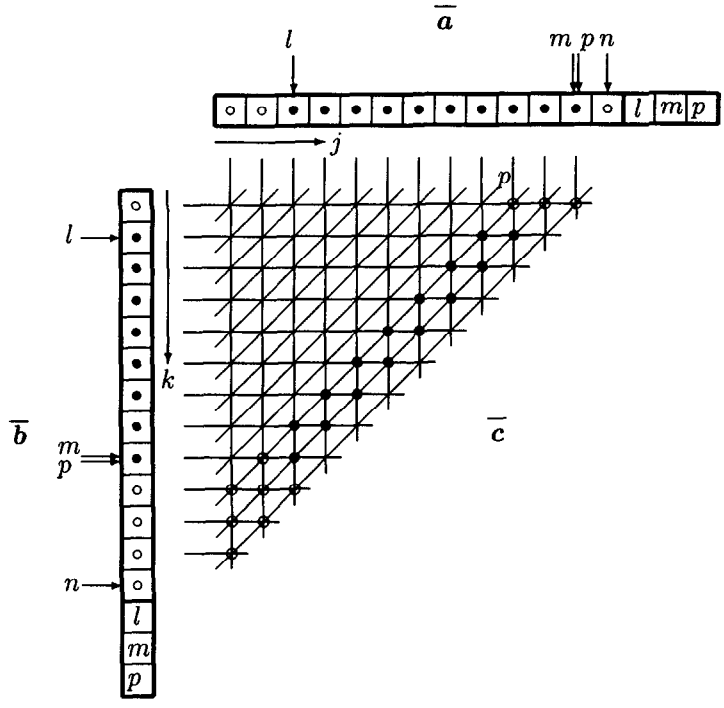


Figure 7. Pointer convolution number multiplication $\overline{a} * \overline{b} = \overline{c}$.

In the differentiation routine, the pointer is shifted one position back and in the integration routine it is shifted one position forward,

$$p_{u'} = p_u - 1, \tag{92}$$

$$p_{u'} = p_u + 1. \tag{93}$$

In each pointer routine, there should be a warning message like "null operation" if the old pointer of the result is already at the new determined value, and an error message should be programmed

if it is higher. Any routine that advances the pointer should have a message when the number is "full", pointer at n, and a routine that puts the pointer back should have a message when the number is "empty", pointer at -1; e.g., the differentiation of a scalar constant represented by convolution number $\{c, 0, \ldots\}$ with pointer $p = 0$ becomes $\{0, 0, \ldots\}$ and the pointer moves to -1. Convolution constants have the pointer p set at n and new numbers that have no element computed yet are called virgin numbers and have the pointer p set at -1. Before a recursive solution loop starts, all pointer numbers must have their pointers set, preferably also in a subroutine to make the main routine well readable. All the convolution arithmetic and integration and differentiation routines are converted to pointer routines by changing the lower and upper boundaries for the new elements, which then constitute a pointer convolution number library.

The purpose of the pointer is primarily to ensure that no higher element is computed which depends on a yet unknown element of the operands. At the same time, it is employed to do the minimum amount of computing in the determination of a new element by not determining a previously computed element over again. As a result, schemes with the pointer convolution number require large storage in computer memory because each elementary operation requires the previous elements and pointer, therefore, all intermediate variables must be stored separately, with separate variable names. As an example, if $\overline{f} = \sqrt{\overline{a} + \overline{b} * \overline{u} + \overline{c} * \overline{u}^{*2}}$ is to be calculated from a new pointer setting in $\overline{u}$, then first of all the polynomial is put into the form of the Horner scheme to use as few as possible multiplications, which results in as few as possible intermediate variables $\overline{v}_i$, which are assigned as shown diagrammatically in the equation below

$$\overline{f} = \sqrt{\overline{a} + \overline{u} * \Big(\underbrace{\overline{b} + \underbrace{\overline{u} * \overline{c}}_{\overline{v}_1}}_{\overline{v}_2}\Big)_{\overline{v}_3}}_{\overline{v}_4}$$

The program steps are

$$\begin{aligned}
\overline{v}_1 &= \overline{c} * \overline{u} && \text{(pointer in } \overline{v}_1 \text{ is reset)}\\
\overline{v}_2 &= \overline{b} + \overline{v} && \text{(pointer in } \overline{v}_2 \text{ is reset)}\\
\overline{v}_3 &= \overline{u} * \overline{v}_2 && \text{(pointer in } \overline{v}_3 \text{ is reset)}\\
\overline{v}_4 &= \overline{a} + \overline{v}_3 && \text{(pointer in } \overline{v}_4 \text{ is reset)}\\
\overline{f} &= \sqrt{\overline{v}_4} && \text{(pointer in } \overline{f} \text{ is reset)}
\end{aligned}$$

At the end of this sequence of equations, the pointers in all variables have advanced the same number of positions and will be in the same position as the pointer in $\overline{u}$, modified by shifts due to leading zeros. If the equations are not purely algebraic but contain integration and differentiation steps, then there is another corresponding shift of pointers. This is the property that allows the programming of the recursive solution of differential equations in convolution numbers.

The pointer numbers require that the program steps are written in mathstyle, i.e., each variable has a different name and cannot be overwritten by another variable with the same name. All the numerical information in the intermediate variables is needed in the next run of the loop with another pointer advance. Special routines may be designed to save intermediate variables, e.g., the addition of a pointer number can be incorporated in the previous multiplication, and addition of several pointer numbers can be done in one routine. This may be an important consideration

in multivariate pointer numbers but is outside the scope of this treatise. In any case, the user of the pointer convolution number routines must write the programs in mathstyle.

The pointer convolution number routines cannot be used like fixed length convolution number routines with overwriting because the initial and final pointers are in different numbers, and therefore a complete separate set of pointer routines must be programmed.

14.1. Modification of Theorems

The several reasons that the element u_j is shifted away from the position j in the convolution function $\overline{f}(\overline{u})$ are the presence of zero leading elements and integration and differentiation. The element indices in the compatibility equations and their conclusions in Theorems 2–4 must be modified accordingly. The index 0 in the theorems must be replaced by leading element l_f in f and l_u in u, i.e., in Theorem 2 the leading element f_{l_f} is a function of the leading element u_{l_u} only. The statement $f_0(u_0) = f(u(0), 0)$ is still true but trivial for a singular convolution number $\overline{u}$. This statement cannot be replaced by a simple statement concerning $f_{l_f}(u_{l_u})$ because it depends on the particular function. The particular solution routine will determine the position l_u and l_f. The leading function element f_{l_f} may still be nonlinear, but it may also be linear in the leading element u_{l_u} even if f is a nonlinear function in $u(x)$. However, once the leading element positions are known, the conclusion of Theorem 4 is still correct if the number l_f is added to the index of f and the number l_u is added to the index of u. This is the theorem that proves that once the leading element is found, the remaining elements are determined recursively.

15. SOLUTION OF NONLINEAR CONVOLUTION EQUATIONS

To solve the nonlinear convolution equation $\overline{f}(\overline{u}) = \overline{0}$ we use the theory of Section 11. The part that remains to be discussed is the numerical determination of the functions f_i in equations (77), (79) etc.

Often, programs are developed having paper work in mind, and with today's computers with large speed and large memory it may be advantageous to similarly use large storage in developing programs to avoid confusion and to make a program readable. Programs like this are versatile and can be changed or adapted easily. In the end, this leads faster to a correct solution. Storage saving and special methods to save computing time make programs unreadable and rigid, needing a large amount of documentation. One structural difference between computer and paper numerical work is the largely exploited custom of computer programming to overwrite variables, i.e., to discard numerical values if they are not used in the subsequent computations, to save storage space, i.e., computer memory, and unfortunately also to get away with less variable names than the programmer cares to invent. In the past, this has mainly been done because of limited computer memory. In paper work, the paper is the storage and exists all the time (unless it is discarded and more often than one would like, retrieved from the waste paper basket again). Paper work therefore corresponds to saving all previous values in memory. Correspondingly, there are two extreme cases of programming the solution routine for nonlinear convolution equations. One method corresponds to paperwork, keeping all intermediate values in storage, needing a large amount of storage. The other method overwrites variables and needs much less storage, but it uses much larger computing time because it has to compute all lost values over and over again.

The recurrence schemes of solutions can be implemented as will be shown with either the fixed length convolution numbers wasting a large amount of computer time, or with truncated numbers saving as much computer time as possible without extra storage, or with pointer numbers using the minimum amount of computer time by storing all intermediate results with the aid of intermediate variables requiring much additional computer memory. This latter is the method that corresponds to paper work.

To explain the method we demonstrate the solution of the quadratic convolution equation of Example 2 of Section 5 again, written as

$$\begin{aligned}\overline{f} &= \overline{b} - \overline{c} * \overline{u} - \overline{\delta}_1 * \overline{u}^{*2} \\ &= \overline{b} - \overline{u} * (\overline{c} + \overline{u} * \overline{\delta}_1),\end{aligned}$$

where $\overline{b}$ is given by equation (64), $\overline{c} = \left\{2, -2\hat{c}, 0, \ldots\right\}$, and treating it as a general nonlinear equation $\overline{f}(\overline{u}) = \overline{0}$ to be solved for $\overline{u}$. The leading element is found by Theorem 3

$$b_0 - c_0 u_0 = 0,$$

from which follows that $u_0 = 0$. This is a single isolated root and, therefore, only one convolution solution follows according to Theorem 5. To determine g_0, we take the convolution derivative

$$\overline{g} \equiv d\overline{f}/d\overline{u} = -\overline{c} - 2\overline{\delta}_1 * \overline{u}$$

evaluated for the zero-element which gives

$$g_0 = -c_0 = -2.$$

1. The simplest method is to set the initial value of the convolution number $\overline{u} = \{u_0, 0, \ldots\}$, set a convolution number $\overline{f}$ empty, and then to proceed to compute $\overline{f}$ with convolution algebra which we write in computerstyle, overwriting the previous variable with a new one,

$$\begin{aligned}\overline{f} &=: \overline{u} * \overline{\delta}_1 \\ \overline{f} &=: \overline{c} + \overline{f} \\ \overline{f} &=: \overline{b} - \overline{f}\end{aligned}$$

According to Theorem 4, the element 1 in $\overline{f}$ is now $f_1(u_0, 0)$, and from equation (78)

$$u_1 = -f_1/g_0.$$

All the higher elements in $\overline{f}$ are useless because they were computed with the wrong convolution number for this purpose. Insert this element in $\overline{u}$, therefore, at this stage, $\overline{u} = \{u_0, u_1 \ldots\}$ and then simply repeat the sequence of equations above. According to Theorem 4 the element 2 in $\overline{f}$ is now $f_2(u_0, u_1, 0)$ and from equation (78)

$$u_2 = -f_2/g_0.$$

Again, the higher elements in $\overline{f}$ are useless because they were computed with the wrong convolution number for this purpose, but the element f_1 is now zero because it is computed with the correct number of elements in $\overline{u}$ to be the correct element $f_1(u_0, u_1)$ of $\overline{f}$ which should be $\overline{0}$.

The solution loop is therefore:

compute u_0

compute g_0

Insert element u_0 in $\overline{u}$

for $i = 1$ to n

$$\overline{f} =: \overline{u} * \overline{\delta}_1$$
$$\overline{f} =: \overline{c} + \overline{f}$$
$$\overline{f} =: \overline{b} - \overline{f}$$
$$u_i = -f_i/g_0$$
$$\overline{u} =: \overline{u} \text{ with element } u_i \text{ added}$$
next i

There are two ways computer time is wasted in this method. One is that all the previous elements in each step are calculated over and over, but another one is that the number $\overline{f}$ is computed to the full length n everytime although only the elements $\leq i$ are needed.

2. The second problem can be eliminated by using truncated convolution numbers in the same routine.

15.1. Pointer Solution of Nonlinear Equations

3. The third method uses pointer numbers and intermediate variables, wasting no computation steps, (the paper method). The recurrence scheme is, using pointer numbers:

Set virgin pointer convolution numbers $\overline{v}_1$, $\overline{v}_2$, $\overline{f}$,

compute u_0
compute g_0
Insert element u_0 in $\overline{u}$
for $i = 1$ to n
reset $p = i$ in $\overline{u}$
$\overline{v}_1 = \overline{u} * \overline{\delta}_1$
$\overline{v}_2 = \overline{c} + \overline{v}_1$
$\overline{f} = \overline{b} - \overline{v}_2$
$u_i = -f_i/g_0$
(contd. next column)

(contd.)
insert element u_i in $\overline{u}$
reset $p = i - 1$ in $\overline{v}_1$
reset $p = i - 1$ in $\overline{v}_2$
reset $p = i - 1$ in $\overline{f}$
next i

Pointer convolution loops are written in mathstyle again because each variable has its own unique meaning. In this method, of course, the pointer is always at the wrong place with respect to the true convolution number which is used for the next loop, but only the last element is incorrect, and this is overwritten by the correct element by means of the resetting of the pointer in the intermediate and final variables. The process of inserting an element into a convolution number in a loop might well be done with the aid of a subroutine where it will automatically be checked against contradictions which could perhaps be missed by the programmer.

A general remark is in order for these recurrence schemes. An element arithmetic is used instead of convolution number algebra, analog to machine programming of digits in a computer instead of using algebraic number routines. Yet the convolution algebra developed makes it possible to use a general numeric method for all nonlinear convolution equations in the loop without symbolic development of the recurrence schemes, which symbolically are all different for each problem and need a lot of detail planning, [27,44]. The recurrence routines are all executed numerically, dynamically programmed as it were, we don't even need expansion formulas like the Binomial Theorem any more.

The method becomes more complicated for cases with multiple roots u_0. The functions containing the multiple roots u_1 can be computed as before, but usually a polynomial routine is available that uses coefficients as input and, therefore, the explicit formulas for the coefficients

are determined as shown below for up to third order. Taking derivatives in the number domain,

$$
\begin{aligned}
&f(u,x)=0,\\
&f' \;=\; 0=\frac{\partial f}{dx}+\frac{\partial f}{du}u'\\
&\qquad\equiv f_x+gu',\\
&f'' \;=\; 0=\frac{\partial^2 f}{dx^2}+2\frac{\partial}{dx}\frac{\partial f}{du}u'+\frac{\partial^2 f}{du^2}u'^2+\frac{\partial f}{du}u''\\
&\qquad\equiv f_{xx}+2g_xu'+hu'^2+gu'',\\
&f''' \;=\; 0=\frac{\partial^3 f}{dx^3}+3\frac{\partial^2}{dx^2}\frac{\partial f}{du}u'+3\frac{\partial}{dx}\frac{\partial^2 f}{du^2}u'^2+\frac{\partial^3 f}{du^3}u'^3+3\frac{d}{dx}\frac{\partial g}{du}u''+\frac{\partial f}{du}u'''\\
&\qquad\equiv f_{xxx}+3g_{xx}u'+3h_xu'^2+lu'^3+3g'u''+gu''',
\end{aligned}
$$

where $g=\dfrac{\partial f}{du}$, $h=\dfrac{\partial g}{du}$, $l=\dfrac{\partial h}{du}$.

Writing the Taylor expansion of each function, and of the total derivatives, then inserting $x=0$ becomes an operation where $\phi^{(k)}(x)$ is converted to $k!\phi_k$. Applying this operation to the equations above transforms them to

$$
\begin{aligned}
f_1 &= f_{x0}+g_0u_1,\\
2f_2 &= f_{xx0}+2g_{x0}u_1+h_0u_1^2+2g_0u_2,\\
6f_3 &= f_{xxx0}+3g_{xx0}u_1+3h_{x0}u_1^2+l_0u_1^3+6g_1u_2+6g_0u_3.
\end{aligned}
$$

These equations are used for the polynomial equations in u_1 that we need in the following cases:

if $g_0=0$ then

f_{x0} must be compatible, i.e., $=0$, and

$$f_{xx0}+2g_{x0}u_1+f_{xx0}u_1^2=0 \tag{94}$$

if also $g_1=0$ then

f_{xx0} and g_{x0} must be compatible, i.e., $=0$, and

$$f_{xxx0}+3g_{xx0}u_1+3h_{x0}u_1^2+l_0u_1^3=0 \tag{95}$$

Equations (94) and (95) are the polynomials in u_1 whose roots are the possible different values of u_1 if the compatibility conditions are met. The different coefficients can be evaluated easily by applying the following rules. g_0, h_0, l_0 are the function values of $g(u,x)$, $h(u,x)$, $l(u,x)$ when $u=u_0$ and $x=0$ which are evaluated by putting u_0 in the place of the argument u and the Taylor coefficients c_{i0} in the place of the functional coefficients into the functions, which becomes the a numerical function evaluation. The Taylor coefficients of the functional coefficients are available anyway because they are the convolution coefficients that are used in the convolution algebra in the solution recurrence scheme. To obtain the values of f_{x0}, g_{x0}, h_{x0}, the functions must be differentiated with respect to their k functional coefficients $c_i(x)$,

$$f_x=\sum_{i=1}^{k}\frac{\partial f}{dc_i}\,c_i' \tag{96}$$

and similarly for the other partial derivatives. Then the functions are evaluated by putting u_0 in the place of the argument u, and the Taylor coefficients c_{i0} and c_{i1} in the place of the functional coefficients c_i and c_i' respectively. Note that we don't need to know what functions the derivatives $c'(x)$ are, we only need the Taylor coefficients of $c(x)$. Similarly, f_{xx0}, g_{xx0} are the function values

of the differentiated functions

$$f_{xx} = \sum_{i=1}^{k} \frac{\partial f_x}{dc_i}\, c_i' + \sum_{i=1}^{k} \frac{\partial f_x}{dc_i'}\, c_i'', \tag{97}$$

which are evaluated by putting u_0 in the place of the argument u and the Taylor coefficients c_{i0}, c_{i1} and $2c_{i2}$ in the place of the functional coefficients c_i, c_i' and c_i'' respectively. Continuing this procedure, f_{xxx0} is the function value of $f_{xxx}(u,x)$ which is evaluated by putting u_0 in the place of the argument u and the Taylor coefficients $c_{i0}, c_{i1}, 2c_{i2}$ and $6c_{i3}$ in the place of the functional coefficients. Also, note that we have used the term functional coefficients for any functional parameters.

15.2. Advance Pointer Convolution Number

The advancement of the pointer and the single element programming in the previous routines distracts from the convolution number concept like machine programming from algebraic number programming. A method is therefore desirable that executes the required steps by a defined operation on the convolution number, even though such an operation does not have an inverse transform that is a well-known or meaningful number operation.

To facilitate the solution of nonlinear equations, particularly where $g_0 = 0$, an advance pointer a is introduced which is a director to execute the pointer settings for the general nonlinear solution

$$u_{i+1} = f_{i+a}(u_0, u_1, u_2, \ldots u_i, 0, \ldots)/g_{a-1}, \tag{98}$$

as discussed in this Section 15.1. The advance pointer a can either be inserted in the argument list of each arithmetical subroutine or it can be appended to the other pointers as shown in Figure 8 in the storage of the convolution number in an array of length $n+4$ (counted from zero). From 0 to p the correct convolution number is stored, which is not going to be changed as the solution proceeds, and from $p+1$ to $p+a$ the required additional information is stored, which is overwritten as the solution proceeds.

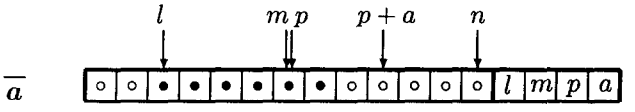


Figure 8. Advance pointer convolution number $\overline{a}$.

In any case, the pointer subroutines are used as before and the advancing and resetting of the pointer p is added at the beginning and the end, respectively. The equations for the solution are written as before in Section 15.1, with the following alterations:

First of all the leading element u_0 of the solution $\overline{u}$ is determined as before as the root of f_0, with possible multiplicity r. The advance pointer is set at the value $a = k+1$ corresponding to the position k of the leading nonzero element g_k, i.e., $a = 1$ for the simplest case $g_0 \neq 0$. If the compatibility equations $f_1 = \cdots = f_{a-1} = 0$ are satisfied, the possible different elements u_1 are determined from $f_a^{[-1]}$ and the convolution number $\overline{f}$ is set at $\overline{0}$ with $l = m = p = a-1$. The convolution number $\overline{\delta}_g$ is defined

$$\overline{\delta}_g = g_k \overline{\delta}_k = \{0, 0, \ldots, g_k, 0, \ldots\}, \qquad k = a-1, \tag{99}$$

with $l = m = k$, $p = n$. Note that we don't call this number $\overline{g}$ which would only be consistent with previous notation if $p = 0$. The steps to compute the convolution number $\overline{f}$ are continued

as before, but in each arithmetic subroutine the pointer p is advanced by a positions in the known arguments before the arithmetic operation and reset to the old after the operation. The numbers are therefore always correct up to the pointer p in the main program, with the required additional elements in the positions $p+a$. All numbers with pointer at i are now actually computed with advanced pointer at $i+a$, starting with the unknown $\overline{u}$. Finally, the pointer number $\overline{f}$ is computed up to $f_{i+a}(u_0, u_1, \ldots, u_i, 0, \ldots)$. The step to compute u_{i+1} in equation (98) is now executed with the full convolution number statement

$$\overline{u} = -\overline{f} / \overline{\delta}_g. \tag{100}$$

Since $\overline{\delta}_g$ has only one element, only the one element of $\overline{u}$ is computed. It is inserted in the correct position because $\overline{\delta}_g$ has the correct number of leading zeros which are compatible in division with the originally inserted zeros in $\overline{f}$. The $i-1$ leading elements of $\overline{u}$ are not affected by this division because they are protected by the pointer p. The advanced pointer position in $\overline{\delta}_g$ becomes n so that the advanced pointer position in $\overline{f}$ limits the division operation to the advanced position i. The pointers l and m in $\overline{\delta}_g$ direct the division by the single element g_{a-1}. Note also that all the leading elements of $\overline{f}$ must finally be zero, but they are only approximated within machine accuracy, while the leading $a-1$ elements must be true zeros. The pointer l_f may then be anything between $a-1$ and p which will allow the correct division of equation (100). One more formal step has to be taken, i.e., to set the pointer p in $\overline{u}$ one ahead according to the new determined element u_{i+1}. If this is programmed as an operation on the convolution number $\overline{u}$, then all the statements in the main program loop become full convolution number statements. Of course some steps can be combined into a single routine, but here we want to stress the basic necessary different convolution number operations.

A complete solution of the root of a fourth order convolution polynomial is given in Appendix 6 illustrating the method of the advance pointer convolution number.

The solution methods in this section demonstrates the purpose of the convolution number, to make it as easy as possible for a programmer to find the numeric solution of Taylor coefficients, bypassing the analytical formulas, which in a general case are too long to be practical for numerical application. We have presented the pointers l, m, and p as necessary tools in the routines to carry out the arithmetic operations fairly effectively. By having an algebra available, better computational efficiency, e.g., as reported in [16,17], is a separate issue that is not part of this treatise, and can always be implemented internally in the routines without changing the algebra—similar to the computer use of matrix algebra. The purpose of computation is to obtain correct answers, and to use the computer efficiently; in this order. We can safely assume that no new program gives a correct result at the first run. To have a programming method that is easy to implement leads faster to a correct solution and in the total is therefore more efficient than complicated computer time saving methods. The researcher needs programs that work. Computer efficiency can rather be implemented afterwards by an experienced programmer if the same routine is going to be used repeatedly.

Let us point out the alternatives if the solution of a nonlinear equation would be needed for numerical purposes, i.e., in a digital program. The first would be to iterate with the function everytime a value is needed, let's say 10 iterations applied to 1000 points; this requires 10000 function evaluations. The second alternative is to obtain the convolution solution, which requires the same 10 iterations to find the leading element, but only once. The determination at the 1000 points is then done with the convolution solution, requiring 1010 function evaluations.

We repeat that, at this stage, we have still not discussed how to find the Taylor transform of all possible functions that may occur as functional coefficients in any linear or nonlinear convolution equations.

16. SOLUTION OF IMPLICIT DIFFERENTIAL EQUATIONS

In this section, the recursive solution of the implicit differential equation is presented for which the theory was developed in Section 11. To illustrate the method and the notation, let us take a simple case so that the first steps can easily be solved by hand,

$$f(u'', u', u) = (u'')^2 - 2u'' - u'u - u^2 = 0. \tag{101}$$

Taylor transform from Theorem 6:

$$\overline{f}\left(\overline{u}'', \overline{u}', \overline{u}\right) = \overline{u}''^{*2} - 2\overline{u}'' - \overline{u}' * \overline{u} - \overline{u}^{*2} = \overline{0}. \tag{102}$$

The solution by hand starts with the convolution variables

$$\begin{aligned}
\overline{u} &= \{u0, u_1, u_2, u_3, u_4, \ldots\} \\
\overline{u}' &= \{u_1, 2u_2, 3u_3, 4u_4, \ldots\} \\
\overline{u}'' &= \{2u_2, 6u_3, 12u_4, \}
\end{aligned}$$

then, inserting them into equation (102) and doing the convolution multiplications, we obtain

$$\begin{aligned}
f_0 &= 4u_2^2 - 4u_2 - u_1u_0 - u_0^2, \\
f_1 &= 24u_2u_3 - 12u_3 - 2u_2u_0 - u_1^2 - 2u_0u_1, \\
f_2 &= 48u_2u_4 + 36u_3^2 - 24u_4 - 3u_3u_0 - 2u_2u_1 - u_1u_2 - 2u_0u_2 - u_1^2. \\
&\vdots
\end{aligned} \tag{103}$$

With the initial conditions u_0 and u_1 for this second order differential equation given, the unknown elements are from u_2 upwards. Corresponding to the theory in Section 13, the lowest unknown element, u_2 occurs nonlinear in f_0, similar to the solution method of nonlinear equations, to the second degree, corresponding to the quadratic equation in the highest derivative, allowing two solutions. The higher elements u_3, u_4 etc. occur linear in f_1, f_2 etc., respectively. Their coefficients can be verified from the theory by determining the derivatives

$$\overline{g} = \partial\overline{f}/d\overline{u} = -\overline{u}' - 2\overline{u} = \begin{bmatrix} -u_1 - 2u_0 \\ -2u_2 - 2u_1 \\ -3u_2 - 2u_2 \\ \vdots \end{bmatrix},$$

$$\overline{r} = \partial\overline{f}/d\overline{u}' = -\overline{u} = \begin{bmatrix} -u_0 \\ -u_1 \\ -u_2 \\ \vdots \end{bmatrix},$$

$$\overline{s} = \partial\overline{f}/d\overline{u}'' = 2\overline{u}'' - 2 = \begin{bmatrix} 4u_2 - 2 \\ 12u_3 \\ 24u_2 \\ \vdots \end{bmatrix}.$$

These column vectors introduced into equation (87) produce the Jacobian matrix

$$\frac{d\overline{f}}{d\overline{u}} = \begin{bmatrix} \frac{\partial f_0}{du_0} & \frac{\partial f_0}{du_1} & \frac{\partial f_0}{du_2} & & \\ \frac{\partial f_1}{du_0} & \frac{\partial f_1}{du_1} & \frac{\partial f_1}{du_2} & \frac{\partial f_1}{du_3} & \\ \frac{\partial f_2}{du_0} & \frac{\partial f_2}{du_1} & \frac{\partial f_2}{du_2} & \frac{\partial f_2}{du_3} & \frac{\partial f_2}{du_4} \\ \vdots & & & & \ddots \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c|c|c|c} -u_1 - 2u_0 & -u_0 & 8u_2 - 4 & & & \\ -2u_2 - 2u_1 & -2u_1 - 2u_0 & 24u_3 - 2u_0 & 24u_2 - 12 & & \\ -3u_3 - 2u_2 & -3u_2 - 2u_1 & 48u_4 - 3u_1 - 2u_0 & 72u_3 - 3u_0 & 48u_2 - 24 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right].$$

These derivatives can be checked by direct appropriate partial differentiation of the elements f_k in equation (103). For the solution procedure we are not interested in the details except the diagonal element, which comes from the leading element of the term

$$\binom{2+k}{k}\overline{s},$$

$$\text{which is} \qquad \binom{2+k}{k} s_0 = (k+2)(k+1)\, q_0.$$

The remaining elements u_i are solved by the remaining linear relations and the coefficient of the unknown element. The recurrence scheme is, using pointer numbers:

Set virgin pointer convolution numbers $\overline{v}_1, \overline{v}_2, \overline{v}_3, \overline{v}_4, \overline{f}$
initial values $u_0 = u(0),\ u_1 = u'(0)$
compute u_2 from $f_0(u_0, u_1, u_2) = 0$
compute $s_0 = 4u_2 - 2$
insert elements $\overline{u} = \{u_0, u_1, u_2, 0, \ldots\}$

for $k = 1$ to n
reset $p = k + 2$ *in* $\overline{u}$
$\overline{u}' = d/dx\,\overline{u}$
$\overline{u}'' = d/dx\,\overline{u}'$
$\overline{v}_1 = \overline{u}' + \overline{u}$
$\overline{v}_2 = \overline{u} * \overline{v}_1$
$\overline{v}_3 = \overline{u}'' - 2$
$\overline{v}_4 = \overline{u}'' * \overline{v}_3$
$\overline{f} = \overline{v}_4 - \overline{v}_2$
$u_{k+2} = -f_k/((k+2)(k+1)s_0)$
(contd. next column)

(contd.)
insert u_{k+2} in $\overline{u}$
reset $p = k - 1$ in $\overline{f}$
reset $p =: p - 1$ in $\overline{u}'$
reset $p =: p - 1$ in $\overline{u}''$
reset $p =: p - 1$ in $\overline{v}_1$
reset $p =: p - 1$ in $\overline{v}_2$
reset $p =: p - 1$ in $\overline{v}_3$
reset $p =: p - 1$ in $\overline{v}_4$
next k

The equations in this loop are written in mathstyle again because each variable has its own unique meaning. The pointers are shifted appropriately in the routines. The element k of the

pointer number $\overline{f}$ is used for the functions of incomplete $\overline{u}$ but all previous elements of $\overline{f}$ are the true elements and should be zero if the computation is correct.

A sample run of the program with initial conditions $u_0 = 2$, $u_1 = 1$ gives values $u_2 = 1.822876$, -0.8228756. The series with the first value computed on an IBM PC converges within a radius < 0.9 and with the second value within a radius < 1.3, for which 48 coefficients were used. The estimate of radius of convergence is discussed in Section 19.

From the illustration it is clear what the general routine for the implicit solution of a nonlinear differential equation $f(u)$ of order m is:

The first $m-1$ elements are given by the $m-1$ initial conditions.

The first unknown element u_m must be found from the function f_0.

Determine the partial derivative w.r.t. the highest derivative of u, $q = \dfrac{\partial f}{du^{(m)}}$.

Determine $q_0 = q(0)$.

The pivot element is

$$\text{diagonal element} \quad d_k = \binom{m+k}{k} q_0. \tag{104}$$

The equations for the remaining elements are

$$u_{k+m} = -f_k(u_0, u_1, \ldots, u_{k+m-1}, 0)/d_k, \tag{105}$$

where the function is calculated with convolution algebra using pointers.

The special cases of multiple roots, rendering the pivot element zero, can be treated by extending the method along the same lines as was shown for nonlinear equations.

Finally, the advance pointer method of Section 15.1 can also be adapted to implicit differential equations with an appropriate convolution number $\overline{d}_k$ for the division step $\overline{u} = -\overline{f}/\overline{d}_k$.

17. SOLUTION OF DIFFERENTIAL EQUATIONS—ANALOG DIAGRAM

The analog computer, [45,46], was increasingly used before the spectacular rise of the digital computer, although it is useful in its own right for applications where more qualitative information is sought, like in linear or nonlinear vibrations, [47,48]. The analog computer program consists of a flow diagram with symbols from electronic circuit analysis, with an input and output, and is still a useful tool today to draw a flow diagram of the sequence of steps in the finite difference solution of ordinary differential equations, and although it is an exact digitization of the analog computer program, the analogy doesn't seem to have been recognized. Much more recently, the same type of flow diagrams have surfaced in digital signal processing [49,50], where a digital sequence is passed through a digital operator. In these diagrams, however, each step is coupled to an operation of the z-transform, which is in concept and in purpose the reverse of the Taylor transform (taking $1/z$ as the variable), and so the analogy with the analog computer operations is not easily recognized.

The five elements of the analog computer are devices for addition/subtraction, integration, differentiation, multiplication by a constant and one general element from multiplication to any other operation that is not easily implemented in an analog computer. These are shown in Figure 9, [45,46,51,52], not including the multiple inputs with scalar constant multiplication.

All variables are functions of the single independent variable which is modelled by time in an actual analog computer. In our application, we don't invert the sign in each operation as in the original analog computer.

In this section, we present the use of the analog diagram, which is an exact analog of the analog computer program, as an aid for the solution of explicit differential equations by means of the pointer convolution number.

The equations that can be solved on the analog computer are explicit differential equations, i.e., the highest derivative occurs linear and therefore can be expressed explicitly as function of lower derivatives. As example the differential equation of the previous section, equation (101), can be expressed explicitly, because it is only quadratic, as

$$u'' = 1 \pm \sqrt{1 + u'u + u^2}. \tag{106}$$

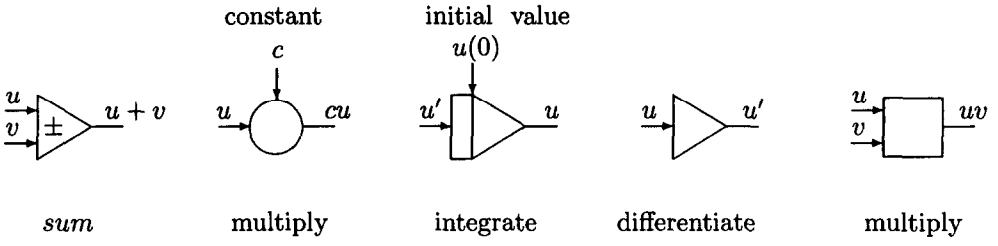


Figure 9. Analog computer elements.

In fact, any implicit differential equation can be expressed as an explicit differential equation, but of one degree higher, by doing one differentiation of the implicit function, e.g., equation (101)

$$f' = \frac{\partial f}{du''} u''' + \frac{\partial f}{du'} u'' + \frac{\partial f}{du} u'.$$

The explicit differential equation of the example is, by differentiating equation (101) and factoring out the highest derivative,

$$u''' = \frac{(u'u + u^2)'}{2u'' - 2}. \tag{107}$$

As a consequence, one initial condition is missing, assuming that the real problem, physical or otherwise, has as many initial conditions as the order of the original form of the differential equation. However, in the method of solution of the implicit differential equation, the first unknown element, u_2 in the previous example, had to be found from a nonlinear equation, before the recursive loop for the remaining elements could proceed. This is then the way the missing extra initial condition for the explicit differential equation of one order higher will have to be determined. Although the equivalence exists, it will be shown that the solution of this equation always involves much more programming and computation than the original nonlinear equation.

For convenience, any explicit differential equation is written in the form

$$u^{(m)} = f(u^{(m-1)}, \ldots, u'', u', u). \tag{108}$$

The analog diagram is the flowsheet of the integration in the convolution domain. The analog diagram starts by integration of the highest derivative, and subsequent integrations that produce all the lower derivatives that are required in the function f in equation (108). This is then fed back into the first integrator. The whole diagram is a loop with input from the outside consisting of the scalar constants or the forcing functions, which in the convolution domain are convolution constants. The analog diagram of equation (106) is shown in Figure 10. Instead of initial value of the real analog computer, the leading element of the output convolution number is supplied at the integrator; this corresponds to the suggested input parameters for the convolution number integration subroutine.

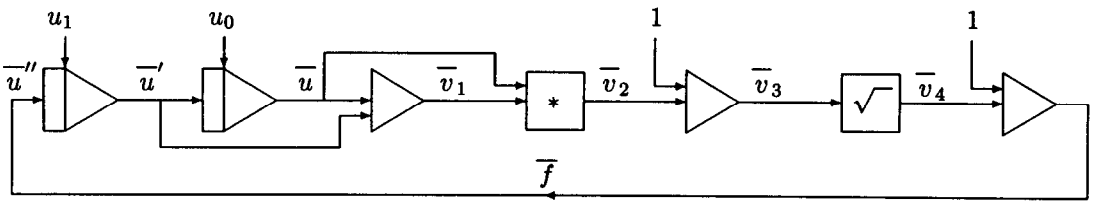


Figure 10. Integration $\overline{u}'' = 1 \pm \sqrt{1 + \overline{u} * \left(\overline{u}' + \overline{u}\right)}$.

The program is written according to the instructions in the analog diagram, written in math-style because pointer numbers are used, with the following steps:

initial values $u_0 = u(0)$, $u_1 = u'(0)$
insert elements $\overline{u} = \{u_0, u_1, 0, \ldots\}$
set initial pointer in $\overline{u}$
set virgin pointer convolution numbers $\overline{u}'$, $\overline{u}''$, $\overline{v}_1$, $\overline{v}_2$, $\overline{v}_3$, $\overline{v}_4$, $\overline{f}$
initial $\overline{u}' = d/dx\,\overline{u}$

for $i = 2$ to n
$\overline{v}_1 = \overline{u}' + \overline{u}$
$\overline{v}_2 = \overline{u} * \overline{v}_1$
$\overline{v}_3 = 1 + \overline{v}_2$
$\overline{v}_4 = \pm\sqrt{\overline{v}_3}$
(contd. next column)

(contd.)
$\overline{u}'' = 1 + \overline{v}_4$
$\overline{u}' = \int \overline{u}''\,dx$
$\overline{u} = \int \overline{u}'\,dx$
next i

In each run of the loop, the function $\overline{f}$ is computed up to the lowest pointer which is contained in its arguments, which is $\overline{u}'$, and the pointer is advanced by one in each integration step. The pointer position for one run of the loop is shown in Figure 11, the intermediate values $\overline{v}_i$ not shown.

The integrations advance the pointers of $\overline{u}'$ and $\overline{u}$ one position forward so that in the next loop one further element of the function $\overline{f}$ is computed. Note that the pointer in $\overline{u}$ is always one ahead, but this value is not used in computing $\overline{f}$ in this loop, only in the next. The minimum amount of computing is done to compute one element in each run of the loop. The convolution number storage in Figure 11 can be placed next to each program step above to follow the computation of new elements and position of the pointer.

The particular feature of this method is that it is completely programmed in terms of convolution numbers, no access to the internal structure of the convolution number is required. However, additional operations may be necessary if shifts due to leading zero elements occur, of which one example is discussed in Section 17.3.

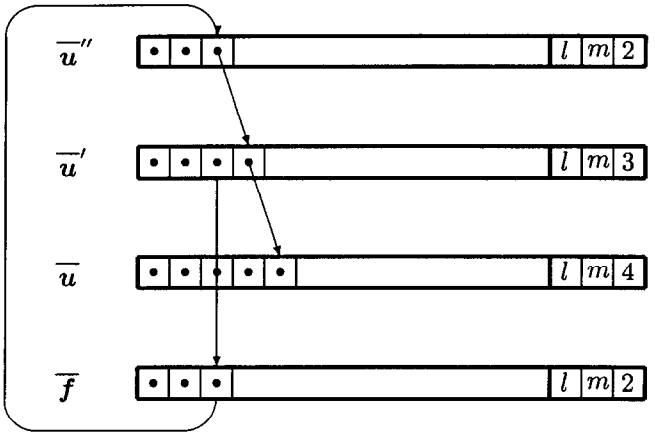


Figure 11. Pointer convolution number integration loop.

Let us compare the program above with the solution procedure of the differential equation of one higher order, equation (107). Following the same rules, the program may be, with new assignments of the intermediate variables,

initial values $u_0 = u(0),\ u_1 = u'(0)$
determine u_2 from nonlinear equation
insert elements $\overline{u} = \{u_0, u_1, u_2, 0, \ldots\}$
set initial pointer in $\overline{u}$
set virgin pointer convolution numbers $\overline{u}'', \overline{u}', \overline{u}, \overline{v}_1, \overline{v}_2, \overline{v}_3, \overline{v}_4, \overline{s}$
initial $\overline{u}' = d/dx\,\overline{u}$
initial $\overline{u}'' = d/dx\,\overline{u}'$

for $i = 2$ to n
$\overline{v}_1 = \overline{u}' + \overline{u}$
$\overline{v}_2 = \overline{v}_1 * \overline{u}$
$\overline{v}_3 = d/dx\,\overline{v}_2$
$\overline{v}_4 = 2\overline{u}''$
$\overline{s} = \overline{v}_4 - 2$
(contd. next column)

(contd.)
$\overline{u}''' = \overline{v}_4 / \overline{s}$
$\overline{u}'' = \int \overline{u}'''\,dx$
$\overline{u}' = \int \overline{u}''\,dx$
$\overline{u} = \int \overline{u}'\,dx$
next i

where we have used as in Section 16, $\overline{s} = 2\overline{u}'' - 2$. Note how much more computation is involved because we divide by the convolution number $\overline{s}$, whereas in Section 15 we only had to divide by the element s_0. More programming is involved because the function $\overline{s}$ is now required in the program. Nevertheless, the implicit nonlinear differential equation can be programmed completely in terms of convolution functions by this means.

17.1. Nonlinear Algebraic Equations

In fact, we may also convert any nonlinear equation from Section 15, $\overline{f}(\overline{u}) = \overline{0}$, to be solved for $\overline{u}$, to a first order differential equation by one differentiation

$$\overline{f}_x + \overline{g}\,(\overline{u}\,)\overline{u}' = \overline{0}$$

$$\text{therefore, } \overline{u}' = -\overline{f}_x / \overline{g},$$

and solve by a loop with pointer numbers, determining first the element u_0 as before. Comparing with the solution in Section 15, we note that the function $\overline{g}(\overline{u})$ must now also be programmed, and a division by the convolution number $\overline{g}$ is required compared to only a division by the element g_0 in Section 15. If one takes a simple example and writes out all the arithmetic operations, it is found that some of the multiplications in compiling f_x are repeated in the division by $\overline{g}$, which consists actually mostly of multiplications according to the division formula in equation (23), to add up to terms which are done by the same multiplication only once in the advance pointer method of Section 15. However, a limitation of the differential equation method is that a multiple root u_0 may not occur, in which case $g_0 = 0$ and the pointer is not moved forward in one run of the loop. Therefore, though it seems that the differential equation method may be preferred because it is simple to program compared to the advance pointer method as illustrated in Appendix 6, there is no other choice if $g_0 = 0$.

17.2. Taylor Series of Functions

All the elements of convolution number analysis were presented separately, which can now be used in combination for a problem whose solution in the form of a Taylor series is required. The following three examples may serve as illustration of the efficiency of the method to solve differential equations. Let us assume that as definition of the function $u(x) = \sin(x)$ the differential equation is used

$$u'' + u = 0, \qquad u(0) = 0, \qquad u'(0) = 1. \tag{109}$$

The analog diagram for the solution of the explicit differential equation $u'' = -u$ is shown in Figure 12.1, where the initial elements of the convolution number $\overline{u}$ are $u_0 = u(0)$, $u_1 = u'(0)$.

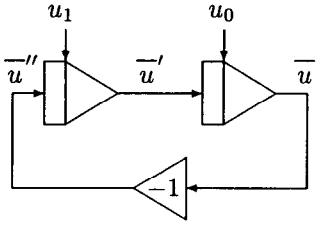


Figure 12.1. Integration $\overline{u}'' = -\overline{u}$.

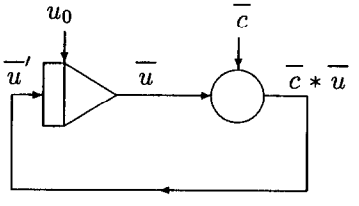


Figure 12.2. Integration $\overline{u}' = \overline{c} * \overline{u}$.

The pointer program is accordingly:

initial values $u_0 = 0,\ u_1 = 1$
insert elements $\overline{u} = \{u_0, u_1, 0, \ldots\}$
set initial pointer in $\overline{u}$
set virgin pointer convolution numbers $\overline{u}', \overline{u}''$,
initial $\overline{u}' = d/dx\overline{u}$
for $i = 3$ to $n + 1$ step 2
$\overline{u}'' = -\overline{u}$
$\overline{u}' = \int \overline{u}'' dx$
$\overline{u} = \int \overline{u}' dx$
next i

The computation steps can be traced according to Figure 11, as being

$$u'_{i+1} = c_{i+1} = \frac{s_i}{i+1}, \qquad u_{i+2} = s_{i+2} = \frac{c_{i+1}}{i+2},$$

which are the recurrence relations for sin and cos Taylor coefficients computed in the most efficient way. The pointer is advanced two positions forward in each loop, a fact that the programmer may have missed, but the integration routine should write the message and exit the loop when the number is full. This use of the pointer relieves the programmer of the task of predicting the number of required loops, which may become a formidable task in long equations that the programmer should not be concerned with. Note that the example problem of analog diagram Figure 10 also contains two successive integrations, but the pointer was advanced by one position in each loop only. The automatic advance of the pointer according to the problem is therefore a very useful feature, relieving the programmer of the detailed internal analysis, he can always find out from the program when the convolution number is full. This example illustrates the well-known fact that Taylor coefficients are quite efficiently developed from a differential equation. This property is widely employed to develop algebraic routines, [27,44], for which we take the general exponentiation $v(x) = u^{\alpha}(x)$, where $u(x)$ is given, as example. This is the method of J. C. P. Miller [8,27], which is repeated here in terms of the method of convolution numbers. If α is not an integer, this is of course not an algebraic routine anymore. The differential equation is constructed by differentiating the nonlinear function and expressing it again in terms of the same variables

$$v' = \alpha v \frac{u'}{u}, \tag{110}$$

which, from the point of view of this section, must be taken as the definition of the function $u^{\alpha}(x)$.

The analog diagram for the solution is shown in Figure 12.2, where the initial element of the convolution number $\overline{v}$ is the function $f(u_0) = u_0^{\alpha}$, and the constant convolution number is first determined with ordinary full length convolution numbers

$$\overline{u}' = d/dx\,\overline{u}, \quad \overline{c} = \alpha \overline{u}' / \overline{u}.$$

The pointer program is:

initial value $v_0 = u_0^{\alpha}$
insert element $\overline{v} = \{v_0, 0, \ldots\}$
set initial pointer in $\overline{v}$
set virgin pointer convolution number $\overline{v}'$

for $i = 1$ to n
$\overline{v}' = \overline{c} * \overline{v}$
$\overline{v} = \int \overline{v}' dx$
next i

Following the single steps, it can be seen that the pointer is not advanced if $\overline{u}$ contains leading zeros, consistent with the fact that generally no Taylor series for $u^{\alpha}(x)$ exists. For integer or even rational values of α special routines can be written to accomodate leading zeros.

Another example is the Jacobian elliptic function which is defined by the elliptic integral

$$F(\phi, k) = u(t, k) = \int_0 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}.$$

The reverse of the differential equation defines the Jacobian elliptic function $t = sn(u)$, of which the Taylor transform is the convolution equation

$$\overline{t}'_k = \sqrt{\left(1 - \overline{t}^{*2}\right) * \left(1 - k^2\overline{t}^{*2}\right)},$$

which can be solved easily with pointer numbers. This method, combined with Taylor series evaluation by Horner scheme, can easily be used in a digital computer subroutine for $sn(u)$ and requires less effort than programming the well-known explicit analytic series expansion [5]. Although other methods may be more efficient and accurate, their programming effort [6], is not worth while except for special applications, bearing in mind that any similar differential equations that occur in the applications may be solved with the same ease. The direct numerical solution of the Taylor series coefficients is in any case always more efficient than reduction to a combination of a large number of standard functions, within the numerical radius of convergence. Most of the required range in the applications, i.e., as required for gyroscope motions and tabulated in [53], can be covered with Taylor series, while at the singular points a combination with other methods may be used.

17.3. Other Operators

The solution of differential equations by pointer numbers as shown by the corresponding analog diagram is based on the property of integration to shift the pointer forward to the position of the new unknown element, while all the functions required to compute the integrand require the known elements only. However, an equal shift may occur because of compatible division or other operations in the computation of the integrand, resulting in a null operation in one loop. In that case, the method of Section 13 can still be employed, but it may be useful to program a special operation routine to supplement the standard routines, so that the standard method of sequence of operations can be used again.

Consider the differential equation

$$u' + e(x)\frac{u}{x} = f(x), \tag{111}$$

where $e(x)$ is regular and nonzero at $x = 0$. The Taylor transform of equation (111) is the convolution equation

$$\overline{u}' + \overline{e} * \overline{u}/\overline{\delta}_1 = \overline{f}, \tag{112}$$

which cannot be solved by the standard method of integrating

$$\overline{u}' = \overline{f} - \overline{e} * \overline{u}/\overline{\delta}_1.$$

Write equation (112) in the form

$$\overline{u}' + (e_0 + \overline{r}) * \overline{u}/\overline{\delta}_1 = \overline{f}, \tag{113}$$

where e_0 is the leading element of $\overline{e}$ and $\overline{r}$ is the remaining convolution number with nonzero leading element which can therefore be divided, $\overline{r}/\overline{\delta}_1 \equiv \overline{p}$, with which equation (113) can be written

$$\overline{u}' + e_0\overline{u}/\overline{\delta}_1 = \overline{f} - \overline{p} * \overline{u},$$

or in the number domain

$$u' + e_0\frac{u}{x} = f(x) - p(x)u. \tag{114}$$

Write equation (114) in operator form

$$\left(\frac{d}{dx}+\frac{e_0}{x}\right)\frac{u}{x}=f(x)-p(x)u.$$

Particularly applied to each term of the Taylor series of $u(x)$,

$$\left(\frac{d}{dx}+\frac{e_0}{x}\right)x^j=(j+e_0)x^{j-1},$$

and, therefore, the inverse operator

$$L(x^{j-1}) \equiv \frac{1}{\frac{d}{dx}+\frac{e_0}{x}}\, x^{j-1} = \frac{1}{j+e_0}x^j, \tag{115}$$

while the ordinary integration operation is

$$\frac{1}{\frac{d}{dx}}\, x^{j-1} = \frac{1}{j}x^j.$$

The operation L on a convolution number is now simply programmed by a modification of the integration routine, replacing the number j by the number $j+e_0$. The convolution equation (112) can now be solved with the operator L in the place of an integration operator $\int$.

17.4. Finite Difference Equations and Digital Filters

Some operations in digital filter analysis have been described as convolution operations, of which an example is the output sequence $y(n)$ of a digital filter in terms of system's input $x(n)$ and unit-impulse response $h(n)$, [50],

$$y(n)=\sum_{k=-\infty}^{\infty}x(k)\,h(n-k),$$

which is the same as $\overline{y}=\overline{x}*\overline{h}$ in our notation, replacing the lower limit by the practical value 0. The implementation of digital recursive filter operations can also be written and implemented in convolution algebra as shown below.

A typical nonhomogeneous first order finite difference equation is

$$f(i)=u(i)+kf(i-1), \tag{116}$$

where the sequence $u(i)$ is given and the sequence $f(i)$ has to be found; we have replaced the customary index n with i to conform to the solution routine notation in the previous sections. Writing $\overline{f}$ and $\overline{u}$ for the sequences, they are not yet convolution numbers because a function $u(x)=\underline{x}\cdot\overline{u}$ is not involved. In fact, in digital filter theory, $f(i)$ and $u(i)$ represent discrete samples of a continuous function $f(t)$ and $u(t)$, respectively. Nevertheless, the simple shift operation $v(i)=f(i-1)$ is the element operation corresponding to the functional operation $v(x)=xf(x)$ if we consider $\overline{f}$ and $\overline{v}$ to be Taylor transforms of $f(x)$ and $v(x)$, respectively, with the transformation equations $f(x)=\underline{x}\cdot\overline{f}$ and $v(x)=\underline{x}\cdot\overline{v}$. In fact, that is exactly what the customary one-sided z-transform does as tool in digital filter theory, except that the variable x is replaced

by the variable $\frac{1}{z}$ which makes the z-transform a half-infinite Laurent series rather than a Taylor series.

Our Taylor transform is the inverse of the z-transform with an inverted variable. On the other hand the actual Taylor coefficients can also be interpreted as sampled values of a continuous function on the level of the Laplace transform, usually demonstrated to show the development of the Fourier integral from the Fourier series [54]. We can therefore, just for convenience, define the sequences in the the finite difference equation as convolution numbers and write equation (116) in convolution algebra as

$$\overline{f} = \overline{u} + k\overline{f} * \overline{\delta}_1, \tag{117}$$

which is programmed in pointer numbers as defined previously, as

$$\begin{aligned} \overline{v} &= \overline{f} * \overline{\delta}_1, \\ \overline{w} &= k\,\overline{v}, \\ \overline{f} &= \overline{u} + \overline{w}, \end{aligned}$$

where the pointers l and m invoke the same intended shift operation without doing unecessary zero operations. Note also that because of the pointer that was introduced, we are not able to put equation (117) into a single convolution number statement, but in principle, the pointer number routines can be extended to accomodate convolution equations in computerstyle as equation (117). Consider now a more general example of a typical recursive digital filter operation [50],

$$y(n) = \sum_{l=0}^{L} A_l\, x(n-l) - \sum_{k=1}^{K} B_k\, y(n-k),$$

which, written in our notation, becomes

$$f(i) = \sum_{k=0}^{l_a} a_k u(i-k) - \sum_{k=1}^{l_b} b_k f(i-k), \tag{118}$$

which is a single step of the convolution algebra operation

$$\overline{f} = \overline{a} * \overline{u} - \overline{b} * \overline{f} * \overline{\delta}_1, \tag{119}$$

according to Section 3, with upper limits $l_a = L$ and $l_b = K$, and lower limit $m_b = 1$.

The convolution solution is simply

$$\overline{f} = (\,\overline{a} * \overline{u}\,) \,/\, (1 + \overline{b} * \overline{\delta}_1\,), \tag{120}$$

with the sequence of operations $\overline{a} * \overline{u}$, completing the computation of all elements, and then the division, completing all elements. But this doesn't provide for a recursive solution during every time interval Δt as intended in digital filters. The sequence of operations of the digital filter is to go through all the convolution algebraic operations in a sequence for each element at a time—which is exactly what the pointer number routines do. The digital filter program for equation (118) is then the same as the equation (119) in pointer numbers, which implements the same digital filter recursive computation. We may even use equation (120) with pointer numbers, supplying the new element u_i and updating the pointer in $\overline{u}$ for each solution loop, and get practically the same computation remembering that the division routine of Section 3, equation (22) or (23), uses elements of the result.

The digital filter implementation diagram of equation (118), [50], is almost the same as the analog diagram of Section 17 designed for equation (119), where z^{-1} of the digital filter corresponds to our multiplication $* \overline{\delta}_1$, i.e., the shift operation. But the digital filter diagram is much more complicated because the storage of every element of the convolution constants $\overline{a}$ and $\overline{b}$ is part of the diagram, while in convolution algebra the storage of elements in the computer is never shown; i.e., in convolution analog diagrams the algebra but not the arithmetic is shown. A digital filter diagram, rather, represents the hardware for a digital filter, or the machine progamming of a convolution algebra routine.

17.5. Several Variables

Differential equations with several variables can be solved by the same pointer method if the highest derivatives occur in such a way that they can be solved independently in a sequence, e.g., as in $u''' = f(u'', u', u, v'', v', v, \ldots,\ v''' = f(u''', u'', u', u, v'', v', v, \ldots$. The unknown elements occur linearly in the equation to be solved in the solution loop by the same argument as set out in Section 11.4, modified only by the shift of the pointer. If they cannot be solved in a sequence as above then a matrix solution must be programmed as special operation. The remaining operations follow the previous method only that the analog diagram consists of so many interlaced integration loops of each variable. The pointer in the convolution variable $\overline{v}$ may not be the same as in the variable $\overline{u}$, either because their highest derivatives are not the same, or because of operations with singular convolution numbers, but the highest elements still occur linear in the element f_i.

17.6. Boundary Value Problems in ODE

The method of convolution numbers is not suited for boundary value problems, also called two point boundary value problems because the boundary consists of the two end points of a region on the real axis. The solutions of the linear differential equation with constant coefficients is well-known and no new method is necessary. The linear differential equation with functional coefficients, e.g., $u'' + p(x)u' + q(x)u = 0$, boundary conditions $u(x = a) = u_a$, $u(x = b) = u_b$ which has a Taylor series solution $u(x) = \underline{x} \cdot \overline{u}$, can be solved for two independent initial conditions by convolution algebra, and then the solutions can be combined linearly to satisfy the two boundary conditions.

For the boundary value problem of nonlinear differential equations, an iterative finite difference method is well-known by its implementation in FORTRAN [55], based on the theory in [56].

To apply the method to the convolution solution of the Taylor transform $\overline{u}$ of the convolution equation, consider, for example, a fourth order equation

$$L(u) \equiv f(u'''', u''', u'', u', u) = 0, \tag{121}$$

with initial boundary conditions at $x = 0$, written as vector variable,

$$\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} u(0) \\ u'(0) \end{bmatrix} \equiv \overline{a}$$

and end boundary conditions at $x = x_b$

$$\begin{bmatrix} u(x_b) \\ u'(x_b) \end{bmatrix} = \overline{b}.$$

Let the other two required and initially estimated initial conditions at $x = 0$ be

$$\begin{bmatrix} u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}u''(0) \\ \frac{1}{6}u'''(0) \end{bmatrix} \equiv \overline{c}.$$

With these initial values, the convolution equation corresponding to equation (121)

$$\overline{L}(\overline{u}) = \overline{f}\left(\overline{u}'''', \overline{u}''', \overline{u}'', \overline{u}'\,\overline{u}\right) = \overline{0} \tag{122}$$

is solved. For the correction in the iteration process the Jacobian matrix $\dfrac{d\overline{b}}{d\overline{c}}$ is required. But the Taylor series evaluation at $x = x_b$ is

$$\overline{b} = \begin{bmatrix} \underline{x}_b \cdot \overline{u} \\ \underline{x}_b \cdot \overline{u}' \end{bmatrix},$$

so that the Jacobian matrix

$$\frac{d\overline{b}}{d\overline{c}} = \begin{bmatrix} \underline{x}_b \cdot \dfrac{d\overline{u}}{d\overline{c}} \\ \underline{x}_b \cdot \dfrac{d\overline{u}'}{d\overline{c}} \end{bmatrix} = \underline{x}_b \cdot \begin{bmatrix} \dfrac{\partial \overline{u}}{du_2} & \dfrac{\partial \overline{u}}{du_3} \\ \dfrac{\partial \overline{u}'}{du_2} & \dfrac{\partial \overline{u}'}{du_3} \end{bmatrix}. \tag{123}$$

To find the four terms in equation (123), consider the variation of equation (121), in the notation of [36],

$$\delta f = \frac{\delta f}{\delta u''''}\delta u'''' + \frac{\delta f}{\delta u'''}\delta u''' + \frac{\delta f}{\delta u''}\delta u'' + \frac{\delta f}{\delta u'}\delta u' + \frac{\delta f}{\delta u}\delta u = 0, \tag{124}$$

which is a linear functional differential equation in δu, and has four solutions depending linearly on the initial conditions. The Taylor transform of equation (124) is, according to Section 8, the convolution differential of equation (122)

$$\begin{aligned} d\overline{f} = {} & \partial\overline{f}/d\overline{u}'''' * d\overline{u}'''' + \partial\overline{f}/d\overline{u}''' * d\overline{u}''' \\ & + \partial\overline{f}/d\overline{u}'' * d\overline{u}'' + \partial\overline{f}/d\overline{u}' * d\overline{u}' + \partial\overline{f}/d\overline{u} * d\overline{u}. \end{aligned} \tag{125}$$

The required two independent convolution solutions are obtained by solving equation (125), using the presently available solution $\overline{u}$ in the convolution coefficients, with the initial conditions

$$\begin{bmatrix} du_0 \\ du_1 \\ du_2 \\ du_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

These are the two solutions $\dfrac{\partial \overline{u}}{du_2}$ and $\dfrac{\partial \overline{u}}{du_3}$, respectively. During the solution loop of equation (125) the two convolution variables $d\overline{u}'$ are also generated, which represent the terms $\dfrac{\partial \overline{u}'}{du_2}$ and $\dfrac{\partial \overline{u}'}{du_3}$, respectively. With the derivatives available, the Newton Raphson iteration of the solution $\overline{u}$ can proceed. Of course, the less analytic and effective alternative to compute the elements $d\overline{u}$ and $d\overline{u}'$ by numerical variation of equation (122) is also possible.

Any more involved boundary value problem where four algebraic functions of the boundary values have to be satisfied, can be solved by the same method by merely including the additional derivatives in the Jacobian matrix.

18. FUNCTIONS

Functions in the number domain in the classical sense consist of a collection of elementary and higher transcendental functions, and of the so-called special functions. Most functions are results of the solution of a problem posed in the form of a particular equation, i.e., elliptic integrals from the circumference of an ellipse, Legendre funtions, Mathieu functions are solutions of partial differential equations of particular boundary conditions, etc. Any problem whose solution can be reduced to a combination or a series of any of the classical functions is accepted to be an analytical solution. The method is obviously based on a tree structure of previously defined functions that can be computed numerically, in the past by looking up computed values in special function tables. These again were computed mainly by Taylor series. With the digital computer the more efficient method is to obtain the Taylor series of any function directly, bypassing the symbolic analysis. From this perspective, there is no distinction between a solution of an equation and a function. Let us define a function in convolution number theory as a statement by means of a convolution equation whose solution is a convolution number, remembering that some convolution equations don't have solutions. A tree structure of functions is necessary because the equation that defines the function may itself use functions which in turn must have been defined before. In the method of convolution analysis, these may be convolution constants or functions, e.g., if $u(x)$ is the function to be found, then $\sin(x)$ in a defining convolution equation is a convolution constant $\overline{s}$, but $\sin(u(x))$ becomes a convolution function $\sin(\overline{u})$. For the same function, there may be more than one definition and we will have to choose the one that is most efficient for computation. All function definitions must define the function in the entire complex plane.

The tree of functions of x is built up in the following levels. If in all these functions the argument x is replaced by the argument $u(x)$, then they transform to convolution functions. If the argument is x, they transform to convolution constants.

1. At the lowest level are the following types:

Polynomials with number coefficients:

$$p(x) = (1 + 23x - 7x^4).$$

Algebraic expressions of polynomials with number coefficients:

$$c(x) = \frac{p(x)}{q(x)}\sqrt{1 - r^3(x)}.$$

Differential Equations with scalar coefficients:

$$\cos(x), \sin(x), \exp(x).$$

Differential Equations with polynomial coefficients:
Legendre functions, Thebycheff functions.

We do not consider Taylor series, with analytic functions of n as coefficients, as function definition in convolution number theory. Such functions may typically be generated by sampled data systems and are discussed in connection with the z-transform [26]. Of course we may use them as constants, but for our convolution function definition we need a finite form. Before they can qualify as convolution numbers, the Taylor coefficients must be put into a recursive computation formula.

For each of these functions there is a convolution number program by the methods described in the previous sections to compute the convolution number.

2. The results from 1. above can then be used to define a following second level of functions: Algebraic expressions of functions:

$$u(x) = \sqrt{1 - \cos^2(x)},$$

$$u(x) + \sqrt{\cos(x)/u(x)} + \sin(x)u^3(x) = 0.$$

Differential Equations with functional coefficients:

$$a(x)u''' + (b(x)u'' + c(x)u)^2 = 0.$$

For each of these functions the convolution program uses the previous defined functions as convolution constant, which must in turn be computed by its own convolution program in the beginning.

3. A next third level of functions are those in whose definition the argument is any of the functions that are defined in the previous level:

Algebraic expressions of functions:

$$2\sin(u(x)) + \frac{b(u(x))}{\sqrt{1 - a(u(x))}} = 0.$$

Nonlinear differential equations:

$$a(x)u''' + b(u', x)u'' + c(u(x)) = 0.$$

From here on the tree continues indefinitely, and at any finite level a function may be defined by a long but finite sequence of previous functions. The convolution program for such a function uses the convolution program of the previously defined function.

As an example let us take the equation

$$u' = u + \sin(u^\alpha). \tag{126}$$

This equation is only a valid statement if the functions $\sin(x)$ and x^α are defined, which has been done by way of examples in Section 12. In the function $v = u^\alpha$, the argument is now the unknown function u and therefore the procedure in Section 12 must be modified to pointer numbers and intermediate variables. The differential equation in Section 12 used for the definition of $\sin(x)$ must now be converted to a differential equation for the function $f = \sin(v(x))$

$$f_{vv} + f = 0,$$

with the integration steps

$$f = \int f_v \, dv = \int f_v \frac{dv}{dx} dx = \int f_v v' dx,$$

$$f_v = \int f_{vv} \, dv = \int f_{vv} \frac{dv}{dx} dx = \int f_{vv} v' dx.$$

The change to the analog diagram of Figure 12.1 is merely that a multiplication element to multiply by v' must be placed before the integration elements. Strangely enough, to compute the function sin of v the external input to the routine is v'. Nevertheless, the value of v_0 is not neglected because it appears in the initial values f_0 and f_{v0} which must also be supplied. For the circuits of both functions v and f, the required variable u' is available from the main differential equation loop. The combined analog diagram for the solution of equation (126) is shown in Figure 13.

For clarity, the connecting lines between the three circuits are not shown, the connections are between the appropriate points marked by •.

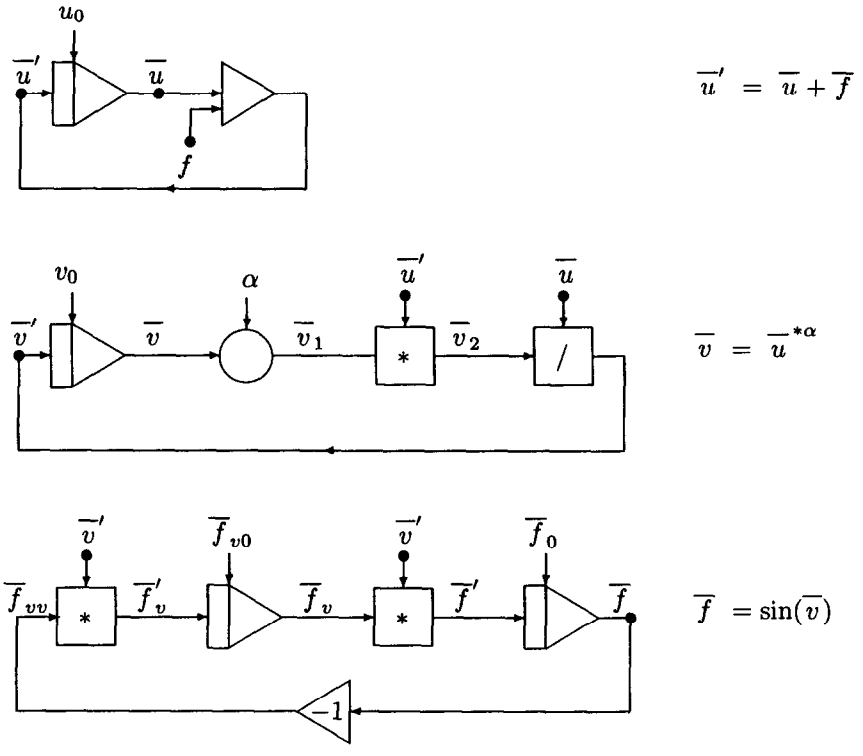


Figure 13. Integration of $\overline{u'} = \overline{u} + \sin(\overline{u}^{*\alpha})$.

The initial value $u_0 = u(0)$ for the upper circuit must be given with the first order differential equation. The initial value v_0 for the second circuit is computed from the equation for the zero element according to Theorem 3, $v_0 = u_0^\alpha$ which we assume is a known scalar formula. Not so with the initial values $f_0 = \sin(v_0)$ and $f_{v0} = \cos(v_0)$ for the last circuit. We have taken equation (109) as an example of how $\sin(x)$ could be defined, therefore, we cannot use any other formula than what comes from this definition which was used to compute the function from the analog diagram Figure 12.1. That computation produced a convolution number which we will now call $\overline{s}$, and we must use this number to compute the Taylor series with the value $x = v_0$ to obtain f_0. We also need the initial value f_{v0} which corresponds to the initial value of the function $\overline{s}'$, which we now call $\overline{c}$, which was not the result of equation (109). Therefore, we must either store the number $\overline{c}$ as well, or we must compute $\overline{c} = d/dx\,\overline{s}$.

The initial values for this problem by the Taylor series are computed by

$$f_0 = \sin(v_0) = \underline{V} \cdot \overline{s}$$

$$f_{v0} = \cos(v_0) = \underline{V} \cdot \overline{c}$$

where $\underline{V}$ = powers of v_0, $\{1\ v_0\ v_0^2\ \ldots\}$ and the actual power series computation is done by Horner's scheme. The pointer number program is:

insert elements u_0 in $\overline{u}$, v_0 in $\overline{v}$, f_0 in $\overline{f}$, f_{v0} in $\overline{f}_v$
set initial pointers in $\overline{u}$, $\overline{v}$, $\overline{f}$, $\overline{f}_v$
set virgin pointer convolution numbers $\overline{u}'$, $\overline{v}'$, $\overline{v}_1$, $\overline{v}_2$, $\overline{f}_{vv}$, $\overline{f}'_v$, $\overline{f}_v$, $\overline{f}'$

for $i = 1$ to n
$\overline{u}' = \overline{u} + \overline{f}$
$\overline{u} = \int \overline{u}'\,dx$
$\overline{v}_1 = \alpha \overline{v}$
$\overline{v}_2 = \overline{u}' * \overline{v}_1$
$\overline{v}' = \overline{v}_2 / \overline{u}$
(contd. next column)

(contd.)
$\overline{v} = \int \overline{v}'\,dx$
$\overline{f}_{vv} = -\overline{f}$
$\overline{f}'_v = \overline{f}_{vv} * \overline{v}$
$\overline{f}_v = \int \overline{f}'_v$
$\overline{f}' = \overline{f}_v * \overline{v}'$
$\overline{f} = \int \overline{f}'$
next i

It doesn't matter whether these equations are written in the correct order, if not, the program may simply report a null operation in the first loop and then continue with the proper pointer settings. This program, with the value $\alpha = 3/2$ is reported in [44], where the same program steps are given which is called the canonical form. The new feature in this treatise is that the recurrence relations don't have to be programmed for each problem and strung together—the program steps already do that with the pointers. In fact, once a function, like $\sin(x)$ in the example, is solved, the steps for $\sin(\overline{v})$ can be programmed as a separate routine, coresponding to the last loop in Figure 13, with a pointer number as argument and a pointer number as result, which is updated at each CALL of the subroutine according to the pointer in the argument. The computation of $\sin(\overline{v})$ then simply takes one line in the above program, which is a CALL of the subroutine. By this means a library of convolution functions that the programmer needs, can be built up with pointer numbers and then used in pointer programs completely similar to programming with ordinary library functions in the number domain. This is an important part of the convolution number theory; it allows us to use defined functions in the convolution number domain with the same ease as in the number domain. The convolution equations can then be programmed as they are written, without clogging the program up with details of function routines. The same as in the number domain applies: the user doesn't even have to know the exact procedure by which the function is programmed. The connection between main program and functions are what is diagrammatically shown in Figure 13 by the •.

Note that Theorem 4 still applies, the elements of the final convolution number are still theoretically exact, consisting of a finite number of algebraic operations of the intial values. If the finite tree of sequence of function definitions becomes too long, then truncation errors accumulate, but still we cannot use the computed Taylor coefficients for the function except for the special argument $\overline{u}$ that has leading zeros, in which case the function $\overline{f}(\overline{u})$ is computed by a Taylor series

$$\overline{f}(\overline{u}) = f_0 + f_1\overline{u} + f_2\overline{u}^{*2} + \cdots$$

programmed in some efficient way, which needs only a finite number of coefficients for the determination of the truncated result. Theorem 4 then still applies.

19. RADIUS OF CONVERGENCE

One aspect that needs attention in the usefulnes of Taylor series solutions is a safe estimate of the radius of convergence. To express any function by means of a Taylor series over a required region on the real x-axis or in the complex plane, the region must be partitioned in overlapping intervals each of which is included in the circle of convergence of a Taylor expansion about a different point. The method is a numerical variation of analytic continuation, which we may call semi-numeric continuation, and consists of computing the value of the function from a first Taylor expansion at an advanced point and then solving the new series with the advanced point as center as reported by Barton *et al.* [44], who have found good results without however relating the advance intervals to the radius of convergence. Particularly, the interval is limited by the radius of convergence in the complex plane, therefore, singularities that may exist in the complex plane quite close to the real line may limit the advance interval size severely. On the other hand, there may be a range with a large radius of convergence and arbitrary small advance step size would be an unnecessary waste of computation, and is still only speculation. Note that the determination of the radius of convergence from the analytic form of the coefficients [7], also called Cauchy-Hadamard formula [8], is not possible.

Therefore, the following procedure is suggested:

Start at beginning of a region.

Solve the equation $l\,(u(x)) = 0$ with initial values to obtain the first Taylor expansion.

Compute the function l that should be zero advancing along the axis until required accuracy limit is exceeded at x_r.

Transform the equation $l\,(u(x))$ to the new variable $x - x_0$, $x_0 < x_r$.

Use function values and derivatives from previous calculation as new initial values and continue.

The method is explained in the following Example 3.

EXAMPLE 3. As example and illustration, we take the solution of the classic Blasius laminar boundary layer equation [57],

$$\begin{aligned} &ff'' + 2f''' = 0, \\ &f(0) = 0, \quad f'(0) = 0, \quad f'(\infty) = 1. \end{aligned} \tag{127}$$

Equation (127) is given in several other normalized forms in [58–60]. This is not strictly an initial value problem but in this particular case the initial value $f''(0)$ can be found by doing one integration with a starting value $f_s''(0) = 1$ until a close enough asymptotic value $f_s(x \to \infty)$ has been reached, then the correct initial value is [59],

$$f''(0) = 1/f_s(\infty)^{3/2}. \tag{128}$$

To serve the purpose of illustration better, equation (127) is written, as in the previous sections, with the notation u for the unknown,

$$l\,(u) = uu'' + 2u''' = 0, \tag{129}$$

and, for the purpose of the solution, the explicit form of the differential equation is

$$u''' = -\frac{1}{2}\,uu''. \tag{130}$$

The convolution number equation to be solved is the Taylor transform of equation (130)

$$\overline{u}''' = -\frac{1}{2}\overline{u} * \overline{u}'', \tag{131}$$

with the leading values $u_0 = 0,\ u_1 = 0,\ u_2 = 1/2$. The analog diagram for the pointer number program is shown in Figure 14.1, from which the pointer program follows

initial values $u_0 = 0,\ u_1 = 0,\ u_0 = 1/2$

set virgin pointer convolution numbers $\overline{v},\ \overline{u},\ \overline{u}',\ \overline{u}''$

for $i = 3$ to n step k

$$\overline{u}'' = \int \overline{u}'''\,dx$$

$$\overline{u}' = \int \overline{u}''\,dx$$

$$\overline{u} = \int \overline{u}'\,dx$$

$$\overline{v} = \overline{u} * \overline{u}$$

$$\overline{u}''' = -1/2\,\overline{u}$$

next i

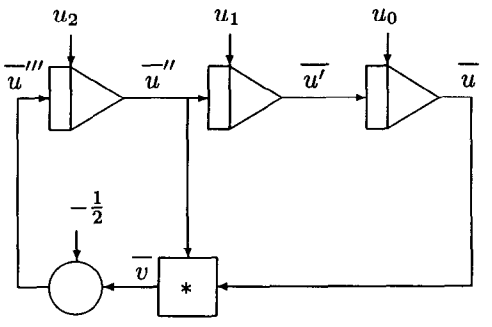


Figure 14.1.

Integration $\overline{u}'' = -1/2\,\overline{u} * \overline{u}''$.

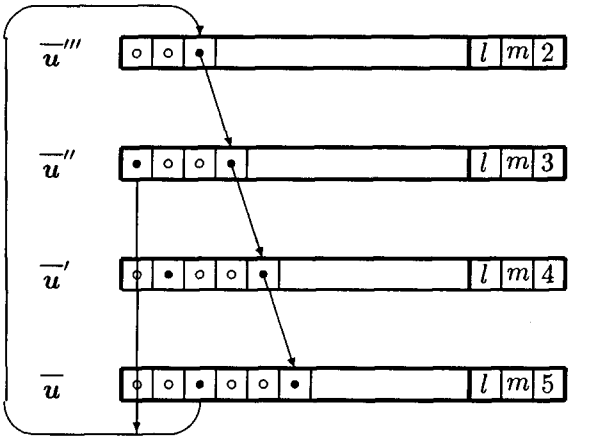


Figure 14.2.

Pointer number integration loop.

It is interesting to follow how the pointer directs the elements to be computed as shown in Figure 14.2, where the pointer values are shown at the end of the first loop, advancing three position each time because one multiplication with two leading zeros occurs. This corresponds to the well-known Taylor series solution where only every third power of x (variable η in the literature) occurs. The efficiency of the pointer technique is apparent, only the required elemets are calculated while the zeros inbetween are just transported. The nonzero coefficients are automatically computed with the efficient recursive sequence. It is not necessary, and indeed undesirable, to program the increment of the loop counter i. The default value of one should be used, and the loop terminated when the pointer number $\overline{u}$ is full after $n/3$ loops, eliminating repetitions with null operationsl. The upper limit n act then only programmeed as a safety measure. But in the continuation regions there are no zero initial values, the convolution numbers are full, and the advance of the pointer is one in each loop only.

Note also that the assumed initial values in the program of this example have not been set. They are automatically inserted in the first run of the loop by the integration routines as described in Section 12.

The function values $u(x)$ are now computed with the Taylor series using the result $\overline{u}$. When advancing along the x-axis there are two reasons for the apparent divergence. One is the theoretical radius of convergence, the other is the result of truncation errors, both of the digital length in the computer and the length of the convolution number. Since the theoretical radius of convergence is unknown, we simply check the apparent radius numerically, which we will call *the numerical radius of convergence.* This numerical limit also exists for entire functions, i.e., that have theoretical infinite radius of convergence.

Consider two different methods. First, we may proceed along the x-axis until a clearly detectable divergence of the function $u(x)$ occurs. This may be not so easy to define because after all the function may have a true steep rise; also if a short truncated convolution number is used the rise may be very gradual, a fact which is particularly exploited in perturbation analysis to approximate function values beyond the radius of convergence. A better measure of divergence is the function $l\,(u(x))$, which is computed along as well and its divergence from zero observed until it exceeds an acceptable error ϵ. This defines the numerical radius of convergence. To be a true analytical representation of the error, we would require a convolution number $\overline{l}$ much longer than the truncated number $\overline{u}$, which is impractical if the largest possible length has already been chosen for $\overline{u}$. A practical method is not to calculate a convolution number $\overline{l}$—which consists of zeros within machine accuracy up to the length n anyway—but in this case, simply to calculate $l\,(x)$ from the evaluated functions $u(x), u'(x), u''(x), u'''(x)$. It may be that this method doesn't always work, especially with linear equations; we have not yet developed a generally proven method. Note that this semi-numeric continuation is different from the analytic continuation [8], inasmuch as we do not transform the Taylor coefficients but use the original defining equation again at every new point of expansion. The method, therefore, does not apply to functions which are defined by Taylor series which we have excluded in Section 18. Actually, we do analytical transformation by indirect means, but only of the first few coefficients equal to the order of the differential equation.

As soon as the numerical radius of convergence has been reached, some slightly smaller radius is chosen and the Taylor series evaluation of the new initial values u, u', u'' is used to start a new computation of the convolution number $\overline{u}$ from equation (131) by the same program. Note however that if functional coefficients are present, which transform to convolution constants, these must also be included in the process—they are not the same convolution constants in another region any more.

The starting solution in the successive regions by this method is shown in Figure 15(a) where the initial solution $u'(x)$ and all the continuations are superimposed on each other, using a convolution number length $n = 50$ on an IBM PC with seven digits. The divergence of the error function $l\,(u(x))$ can be seen clearly, the error function diverging earlier than the solution, which then indicates the numerical radius of convergence. The corresponding circles of convergence are superimposed on the same figure, using the plane of the paper as the complex plane $z = x + \hat{\imath}\, y$. The values are shown in the first two columns of Table 2 until the numerical asymptotic value of $u' = 2.085409$ was reached at $x = 6.1$. On a HP 9836 which uses 13 digits, the asymptotic value of $u' = 2.0854091764$ was reached at $x = 7.6$, using the same convolution number length. From equation (128), this produces the correct initial value of $u''(0) = 0.3320574$ (0.3320573362 on HP 9836). The second computation, with the correct initial value of u'', yields successive numerical radii of convergence as shown in the last two columns Table 2, and they are shown graphically with the divergences in Figure 15(b). The divergences on the other end of the circle of convergence have the same appearance; they are not shown in the figure. Note that the second circle of convergence includes the origin and an approximation of the asymptotic region at $x = 6$.

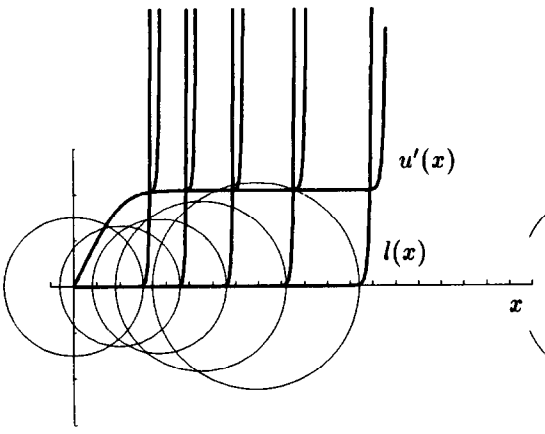


Figure 15a. Blasius starting solution.

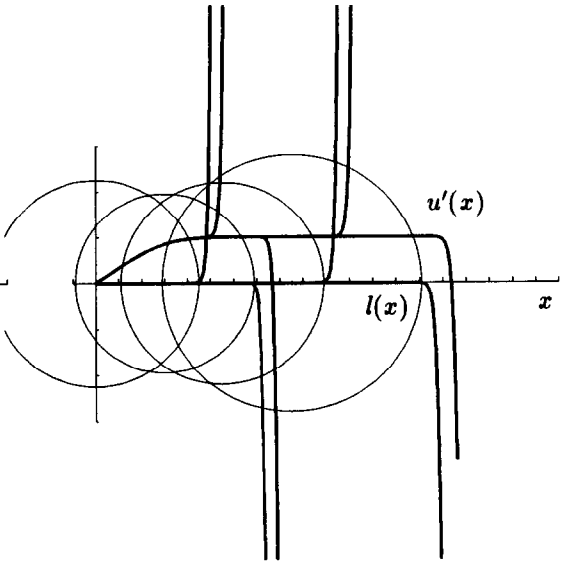


Figure 15b. Blasius final solution.

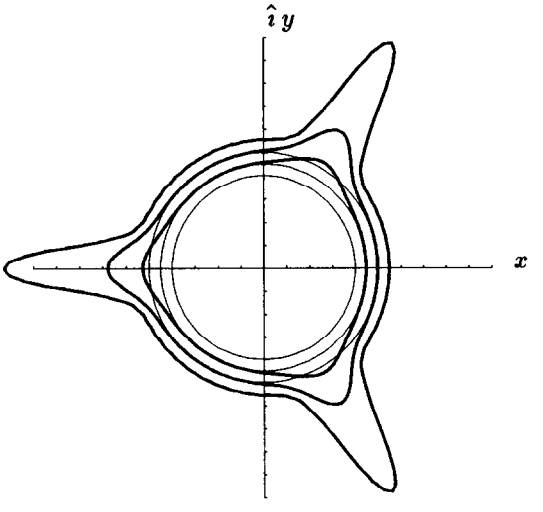


Figure 15c. Blasius complex function.

Table 2. Numerical radius of convergence of Blasius function.

	starting function		final function	
interval	center	radius	center	radius
1	0	3	0	4.5
2	2	2.6	3	3.9
3	3.7	2.9	5.5	4.4
4	5.5	3.7	8.5	5.5
5	7.9	4.5		

The radius of convergence of the solution of a nonlinear equation depends on the solution; the radii in Figure 15(b) can be seen to be much larger than in Figure 15(a). Yet, even in the final solution it is clear that more than one series must be computed to reach the asymptotic value, at $x = 9.1$ (10.9 on HP 9836). There is no problem of continuation by this method with convolution numbers of smaller length, i.e., $n = 12$, or even the minimum of $n = 3$ (always counted from 0), using the same program. The accumulative error of the calculated function will then increase due to the increasing number of continuation intervals, approaching a Finite Difference Method.

In fact, the whole method can now be seen as a kind of n^{th} order Finite Difference Method with adaptive step length, which are the intervals in the previous description, with the feature that the step length has to be probed during computation instead of being predetermined. The microstep length for the probing has to be estimated in relation to the expected continuity of the solution function.

Looking at the circles of convergence in Figure 15(b), there seems to be a singularity in the right half of the complex plane that determines the radii of convergence of all the intervals. Evidence of a singularity is a large value of the function or its derivatives in the neighborhood of the singularity, therefore the abs. function value $u'(z)$ in the complex plane is plotted at varius radii in Figure 15(c), which indicate the presence of three rotationally symmetrically situated singularities 120 deg apart. These are actually found to be lying at $z = 5.69003805e^{\pm\hat{\imath}\pi/3}$, showing the importance of the singularities in the complex plane rather than on the real axis. The details of this and other features are given in Appendix 7.

Of course we cannot distinguish by this method stable and unstable solutions which both satisfy the defining equation and the initial values within machine accuracy, a fact which is very close to the practical behaviour of a corresponding system.

Eventually, every convolution number solution is used as coefficients in the series of the function in number domain applications, typically, we may have a function *blasius(x)* which is used in a program of number computations by a CALL to a subroutine. In this subroutine, the convolution numbers of all intervals and their intervals must be stored, so that to the user it is one continuous function. At every CALL, a Taylor series computation is performed in the subroutine.

The more accurate computations on the HP 9836 were used to compare the function values with the classic published ones in [57] and it was found that in many cases the last digit is slightly in error.

By the same method, other laminar boundary layer ordinary differential equations can be solved, where previously a large effort was made to find the coefficients in analytic form first, e.g., [57,61,62].

A similar remark as in Section 11 applies, that because of the theoretically infinite number of terms required to obtain the exact boundary value at infinity, the final convolution solution does not strictly satisfy the definition of a convolution number in terms of the originally set boundary values.

19.1. Stability

The arithmetic routines of addition/subtraction and multiplication consist of sums that are straight forward and the result is a stable sequence of numbers within digital truncation errors. A stability problem arises in the recursive routines, starting with division. It turns out that the Taylor series of the division and the coefficients, i.e., the convolution number elements, are affected differently.

To demonstrate the situation, we follow the recursive solution steps of the division $\overline{c} / \overline{b} = \overline{a}$ with a small finite convolution number $\overline{b} = \{ b_0, b_1 b_2 \}$. The division consists of solving for the elements of $\overline{a}$ from the multiplication $\overline{a} * \overline{b} = \overline{c}$ which is shown schematically in Figure 16(a) below, from which we take the general equation

$$b_0 a_{i+2} + b_1 a_{i+1} + b_2 a_i = c_{i+2}. \tag{132}$$

Equation (132) is a finite difference equation which has apart from the solution which is developed from the first three lines, an additional solution of the homogeneous equation

$$b_0 h_{i+2} + b_1 h_{i+1} + b_2 h_i = 0. \tag{133}$$

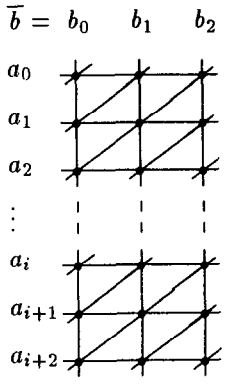


Figure 16a. $\overline{a} * \overline{b}$.

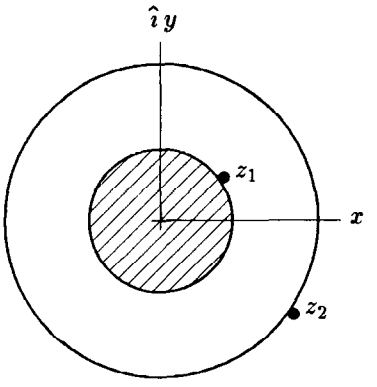


Figure 16b. Singularities of $a(z)$.

The solution of the constant coefficient linear equation is found by the method of assuming $h_i = r^i$, [54], which substituted in equation (133) produces, after cancelling the common factor r^i,

$$b_0 r^2 + b_1 r + b_2 = 0. \tag{134}$$

If we write equation (134) in the form

$$b_2 \left(\frac{1}{r}\right)^2 + b_1 \left(\frac{1}{r}\right) + b_0 = 0, \tag{135}$$

it is clear that the solution is $\frac{1}{r} = z_1, z_2$, where z_1 and z_2 are the roots of the polynomial of the denominator $b_0 + b_1 z + b_2 z^2 = b_2(z - z_1)(z - z_2)$. The general solution of equation (133) is now

$$h_i = \alpha_1 (r_1)^i + \alpha_2 (r_2)^i = \alpha_1 \left(\frac{1}{z_1}\right)^i + \alpha_2 \left(\frac{1}{z_2}\right)^i,$$

so that an additional solution of the Taylor series

$$\alpha_1 \left(\left(\frac{1}{z_1}\right)^i z^i + \left(\frac{1}{z_1}\right)^{i+1} z^{i+1} + \cdots \right) + \alpha_2 \left(\left(\frac{1}{z_2}\right)^i z^i + \left(\frac{1}{z_2}\right)^{i+1} z^{i+1} + \cdots \right)$$

shows up. Starting with the leading lines of the solution for a_i, the factors α_1 and α_2 are zero, but during the following computation it is inevitable that they develop into some small nonzero quantity due to digital truncation errors. The two additional series converge for $z < |z_1|$ and $z < |z_2|$, but diverge for $z > |z_1|$ and $z > |z_1|$. The function $a(z) = c(z)/b(z)$ has singularities at the zeros of the denominator and therefore the true Taylor series $\underline{x} \cdot \overline{a}$ converges only within the radius of convergence given by the lesser value $|z_1|$, which is the hatched area shown in the complex plane in Figure 16(b). In this region, both α-series due to the homogeneous solution also converge, adding small values proportional to α_1 and α_2 to the correct function value, so no harm is done well within the region of convergence. The conclusion is that the recursive solution of division produces a stable result of a Taylor series with only a small α-error which is negligibly small within the region of convergence. Outside it contributes to the already diverging sum.

The situation of the coefficients a_i is different. The α errors appear in any case independent of the use of the Taylor series within the region of convergence, and the α_1 error, due to the root z_1 on the convergence boundary, grows relative to the true coefficients at least as fast as the true coefficients themselves, until they become of the same size or may even completely mask the true, relatively smaller coefficients within the limited digital numbers. This is visually apparent if $|z_1| \ll 1$ and $|z_2| \gg 1$. This feature therefore limits the significant length of the convolution number, which is also clearly displayed in the numerical results.

20. REVERSION OF FUNCTIONS

In terms of the numerical applications that are considered in this treatise, reversion of functions actually means to find the Taylor transform of a function which is defined in a reverse order. The particular classic problem of reversion of a power series [7,8,15,27], is a special case inasmuch as a Taylor series is only a definition of a function in a limited region. In our numerical approach with convolution numbers this is not a special problem, only a particular case of the solution of a nonlinear equation and we treat it here only to show the connection with the classical theory, and to illustrate multivalued solutions, which are not considered in the classical theory, and their radius of convergence. A particular characteristic of the reverse of a single valued function is that the reverse is generally multivalued. The different roots of a nonlinear convolution equation that were discussed in Section 11 were already reversions of functions.

The posed problem is to find for a given polynomial

$$w = w_1 z + w_2 z^2 + \cdots + w_n z^n, \tag{136}$$

the reverse function in the form

$$z = z_1 w + z_2 w^2 + \cdots, \tag{137}$$

which is a Taylor series. In the classic problem of the reversion of a series, the particular feature is that the leading coefficients w_0 and z_0 are missing. i.e., zero. Such a polynomial is called an almost unit in [8]. For that special case, consider two methods for the solution.

Method 1. Write equation (136) as Taylor transformation

$$w = \underline{Z} \cdot \overline{w}, \tag{138}$$

where $\overline{w} = \{0\ w_1\ w_2\ w_3 \ldots\}$, and substitute into equation (137) which is considered as $z = f(w)$, so that, according to convolution Theorem 1,

$$\underline{Z} \cdot \overline{\delta}_1 = \underline{Z} \cdot \left\{ 0 + z_1 * \overline{w} + z_2 * \overline{w}^{*2} + z_3 * \overline{w}^{*3} + \cdots + z_n * \overline{w}^{*n} \right\}. \tag{139}$$

Setting the convolution numbers on both sides of equation (139) equal produces the matrix equation

$$\begin{bmatrix} 0 \\ 1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \overline{\delta}_0 & \overline{w} & \overline{w}^{*2} & \overline{w}^{*3} & \cdots \end{bmatrix} \cdot \overline{z}, \tag{140}$$

where the matrix is known because $\overline{w}$ is given, and $\overline{z} = \{0\ z_1\ z_2\ z_3 \ldots\}$ is the unknown vector of infinite length. The column $\overline{\delta}_0$ has been substituted formally for $\overline{w}^{*0}$. Because of the leading 0 in $\overline{w}$ all powers $\overline{w}^{*j}$ have j leading zeros, counting from 0, therefore, the resulting infinite matrix equation is lower triangular and can be solved by backsubstitution even if $\overline{w}$ is infinite, to any finite number of elements z_j, including the solution $z_0 = 0$, numerically machine exact and analytically exact. Strangely enough, as simple as it is, this is not the classical method.

Method 2. Write equation (137)

$$z = \underline{W} \cdot \overline{z}, \tag{141}$$

where $\overline{z} = \{0\ z_1\ z_2\ z_3 \ldots\}$, and substitute into equation (136), so that

$$\underline{W} \cdot \overline{\delta}_1 = \underline{W} \cdot w(\overline{z}) \tag{142}$$

from convolution Theorem 1 where $w(z)$ is the function of equation (136). This corresponds to the nonlinear convolution equation $f(\overline{z}) = w(\overline{z}) - \overline{\delta}_1 = \overline{0}$ which is discussed in Section 11 and which can be solved by the method in Section 15. Applying this method, the nonlinear computation of the element z_0 is skipped because its value $z_0 = 0$ is already known. Equation (77) of Section 11 is now explicitly, with $g_0 = w_1$,

$$\begin{aligned} f_1 &= 0 = w_1 z_1 - 1, \\ f_2 &= 0 = w_2 z_1^2 + w_1 z_2, \\ f_3 &= 0 = w_3 z_1^2 + 2 w_2 z_1 z_2 + w_1 z_3, \\ &\vdots \end{aligned}$$

The first equation confirms that $z_1 = 1$ if $w_1 = 1$, and from the next ones $z_2, z_3, \ldots$ are computed recursively. This is the classical method [7,8,15,27,63].

Another possibility is the method of Section 17, to differentiate equation (136) w.r.t. w to obtain the differential equation $1 = w_z z'$ and to solve its Taylor transform $\overline{z}' = 1/\overline{w}_z$ with the initial condition $z_0 = 0$ with pointer numbers. This method is in fact treated in [42], where $1/\overline{w}_z$ is expanded analytically (q' = 1 / p'(q) in that notation). As explained in Section 11 this is merely another but numerically less efficient way of the solution method of Section 15.

The particular feature of Method 2 is that if the function $w(z)$ has infinitely many terms, the functions f_i still only have a finite number of terms, up to z^i, because of the leading zero in $\overline{z}$. Therefore, all the elements z_i are computed explicitly and exact from linear relations. This method is given in matrix form in [8], otherwise in explicit equations [7], or in program form [27]. But consider now other solutions $z(w)$ that have $z_0 \neq 0$ leading elements, e.g., the solution with $z_0 = \pi$ of $\arcsin(w)$ when the series $w = \sin(z)$ is given, and the term π is added to the rhs. of equation (137). Method 1 simply fails, whether $w(z)$ is an infinite series or not, equation (140) is not satisfied. This feature can be traced to the fact that the series of equation (136), which is valid also for $z = 0$, is substituted into the modified reverse series $z(w)$ of equation (137), which is only valid for values $\frac{\pi}{2} < z < \frac{3}{2\pi}$, in which $z = 0$ is not embedded, and therefore separation of coefficients of equal powers of z is not valid any more.

Method 2 fails in a different way. For the same example the range $\frac{\pi}{2} < z < \frac{3}{2}\pi$ of the series of equation (137) is embedded in the range of valid z in equation (136), and the substitution of equation (137) into equation (136) is valid. The reverse of a polynomial can be solved this way, but if the series $w(z)$ is infinite the functions f_i now contain infinitely many terms.

From convolution number theory, it is therefore true enough that equation (137) with $z_0 = 0$ is the one and only one [7], being the only one of which elements are determined in a finite number of steps, but in practical applications, first of all, a series with a zero leading coefficient is an exception, and secondly, picking out one of a multivalued solution is a misleading preference. In our convolution number theory, a reversion of a series does not occur because for the same reason a function is not defined by a series.

However, if $w(z)$ is a polynomial then it is an entire function valid in the whole z-plane, and it can be computed machine exact within a large finite radius. Applying Theorem 3, the element z_0 is the solution of the equation $w(z_0) = 0$ which has 0 only as the one easily recognizable root but also $n - 1$ other roots, which lead to $n - 1$ other solutions as well by Theorem 5, which can all be solved by the same Method 2 above.

A polynomial $w(z$ can always be transformed to the singular form of equation (136) by transforming the variable z to the new variable $\zeta = z - z_0$, where z_0 is one of the roots, becoming $p(\zeta)$, and then the classic reversion can be done to find the singular reversion Taylor series $\zeta(p)$. Our purpose here however is to illustrate the reversion of the general regular poynomial.

It is clear that the problem of reversion of a polynomial, considering all the different roots, is but a particular case of the solution of a nonlinear convolution equation. In fact, the more

general case is the equation $f(u) = g(p)$, from which either the expansion $u = \underline{P} \cdot \overline{u}$ or the expansion $p = \underline{U} \cdot \overline{p}$ must be obtained. Example 2, as treated in Section 15, was such a case.

To illustrate the characteristics of a multivalued reversion of a single valued function, we consider the reversion of a polynomial as example.

20.1. Notation of Reversion

We follow [42] to distinguish between reversion and inversion, and use the symbol $p^{[-1]}$ for the reverse function of $p(z)$, but with numerical values in mind we use different variable names for different arguments in the same context. The reverse of the function $w = p(z)$ is then $z = p^{[-1]}(w)$. Particularly, we reject the use of p^{-1}, e.g., writing $\cos^{-1}(x), \sin^{-1}(x)$ instead of $\arccos(x)$ and $\arcsin(x)$, respectively, which is inconsistent with the exponent notation as in $\cos^2(x)/\cos^3(x) = \cos^{-1}(x)$. The relation between the inverse and reverse of a function is that the derivative of the reverse is the inverse of the derivative of the function,

$$\frac{dp^{[-1]}(w)}{dw} = 1/\left(\frac{dp(z)}{dz}\right).$$

20.2. Reversion of Polynomials

We choose an arbitrary simple polynomial of eighth order to illustrate the reversion of an m^{th} order polynomial.

EXAMPLE 4. Let the a polynomial be given

$$w = w_0 + w_1 z + w_2 z^2 + w_3 z^3 + w_4 z^4 + w_5 z^5 + w_6 z^6 + w_7 z^7 + w_8 z^8, \tag{143}$$

where the coefficients w_i are listed in Appendix 8. Note now that the Taylor transform of equation (143) is ambiguous, inasmuch we can either consider w as function of z, equation (143) being an explicit function and the rhs. a convolution constant, or we can consider z as function of w, equation (143) being an implicit function. The problem of reversal can be stated more elaborate as follows. The reverse is required

$$z = z_0 + z_1 w + z_2 w^2 + z_3 w^3 + \cdots \equiv \underline{W} \cdot \overline{z}. \tag{144}$$

The convolution number equation to be solved is obtained by substituting equation (144) into equation (143)

$$\overline{\delta}_1 = w_0 + w_1 \overline{z} + w_2 \overline{z}^{*2} + w_3 \overline{z}^{*3} + w_4 \overline{z}^{*4} + w_5 \overline{z}^{*5} + w_6 \overline{z}^{*6} + w_7 \overline{z}^{*7} + w_8 \overline{z}^{*8}, \tag{145}$$

which can be recognized as one of the two possible Taylor transforms of equation (143). This is a special case of Section 15, of finding a convolution root of a convolution polynomial, even though the only nonscalar convolution coefficient is $w_0 - \overline{\delta}_1$. The eight different leading elements z_0 are found by Theorem 3 from the eight roots of the scalar equation

$$0 = w_0 + w_1 z_0 + w_2 z_0^2 + w_3 z_0^3 + w_4 z_0^4 + w_5 z_0^5 + w_6 z_0^6 + w_7 z_0^7 + w_8 z_0^8. \tag{146}$$

To interpret the eight different roots, the polynomial $w(z) = \phi + \hat{\imath}\psi$ is plotted in the z-plane by constant ϕ-lines and constant ψ-lines, which we call by hydrodynamic analogy potentialines and streamlines, respectively, in Figure 17. The function $w = p(z)$ is single valued and entire, but the reverse $z = p^{[-1]}(w)$ is a multivalued function, eight-valued in this case. The w-plane is shown in Figure 18(a) with the singularities where $\dfrac{dw}{dz} = 0$ and the branchlines are arbitrarily chosen

along streamlines to the left. The branchlines represent cuts in the w-plane through which the eight different Riemann planes are connected on which the function is continuous.

In the w-plane the different Riemann planes and the crossings between them are shown in Figure 18(b) which represents a diagrammatic crossection through the branchlines. The 8 planes have been assigned arbitrary as shown in Figure 18(b), and are connected across the branchlines whose mapping are the thick drawn streamlines in Figure 17.

Once this choice of planes is made, the function values $z(w)$ for each w can be made single valued by defining a reverse function on each Riemann plane, which can be denoted by an index in the reverse function notation, i.e., $z(w,1) = p^{[-1]}(w,1)$ is the reverse function on Riemann plane 1. Each of the eight roots z_0 lie on a particular plane k and the Taylor expansion of the reverse function with this particular root is then the single valued Taylor expansion of $p^{[-1]}(w,k)$. Considering Figure 18(b) as a true geometrical crossection through the Riemann planes, it is obvious that the w-plane in Figure 18(a) is not a true projected view of the Riemann planes, since the branches singularities lie all in different planes. The eight different Riemann planes are shown in Figure 19. In each one of these planes there are some but not all of the singularities, and in each plane k the distance from the origin to the nearest singularity determine the radius of convergence of that particular Taylor expansion of $p^{[-1]}(w,k)$ as is shown in Figure 19. In these figures, the circles of convergence are shown and lines of a mapping function $log(w)$. Without discussing the topology of the connection between the planes, which is outside the scope of this treatise, we note that there are $m-1$ branches that connect m Riemann planes. The $m-1$ singularities lie on the ends of the branches between the planes, so that each singularity is shared by two planes. Each plane must have at least one singularity, but at least one plane must have more singularities so that crossing to more than two planes through the branchline is possible. Particularly, singularities in one plane are not necessarily singularities in any other plane. But whichever way the cuts are chosen, the radius of convergence of the reverse function for each root z_0 is fixed. At least two of them are limited by the same singularity, but others are limited by other singularities. There is even one singularity which does not determine any one radius of convergence.

The corresponding convergence regions in the z-plane are shown in Figure 20 which are the lemniscate-like mappings of the convergence circles. Only in these regions are the reverse Taylor series expanded about the origin $w = 0$ valid. This concludes what we may term an "anatomy of a polynomial."

The computation of the reverse is done choosing the method of Section 17, by differentiating equation (143), rearranging to an explicit differential equation and taking the Taylor transform

$$\overline{z}' = \frac{1}{w_1 + \overline{z} * (2w_2 + \overline{z} * (3w_3 + \overline{z} * (4w_4 + \overline{z} * (5w_5 + \overline{z} * (6w_6 + 7w_7\overline{z}))))) }. \tag{147}$$

The differential equation (147) is solved eight times, each time with another of the scalar roots z_0 as leading elements, with the pointer convolution number method of Section 17, choosing intermediate variables. The results $\overline{z}$ with the length $n = 12$ are listed in Appendix 8, the first element being the root z_0.

The accuracy of the results is illustrated graphically by showing the mapping function $\log(w)$ for each region of convergence in Figure 20, once by the true mapping $z(w)$ and once by the truncated series approximation $\underline{W} \cdot \overline{z}$. The lines are simply plotted on top of each other, and can be distinguished by the smoothness and conforming to the lemniscate-like boundaries of the convergence region of the exact mapping. These graphs, although small in this presentation, give an idea of the region for which the Taylor expansion truncated at $n = 12$ of the reverse function is sufficiently accurate. The eight convolution numbers $\overline{z}$ are the eight convolution roots of the eighth degree convolution polynomial of equation (145) according to Theorem 5.

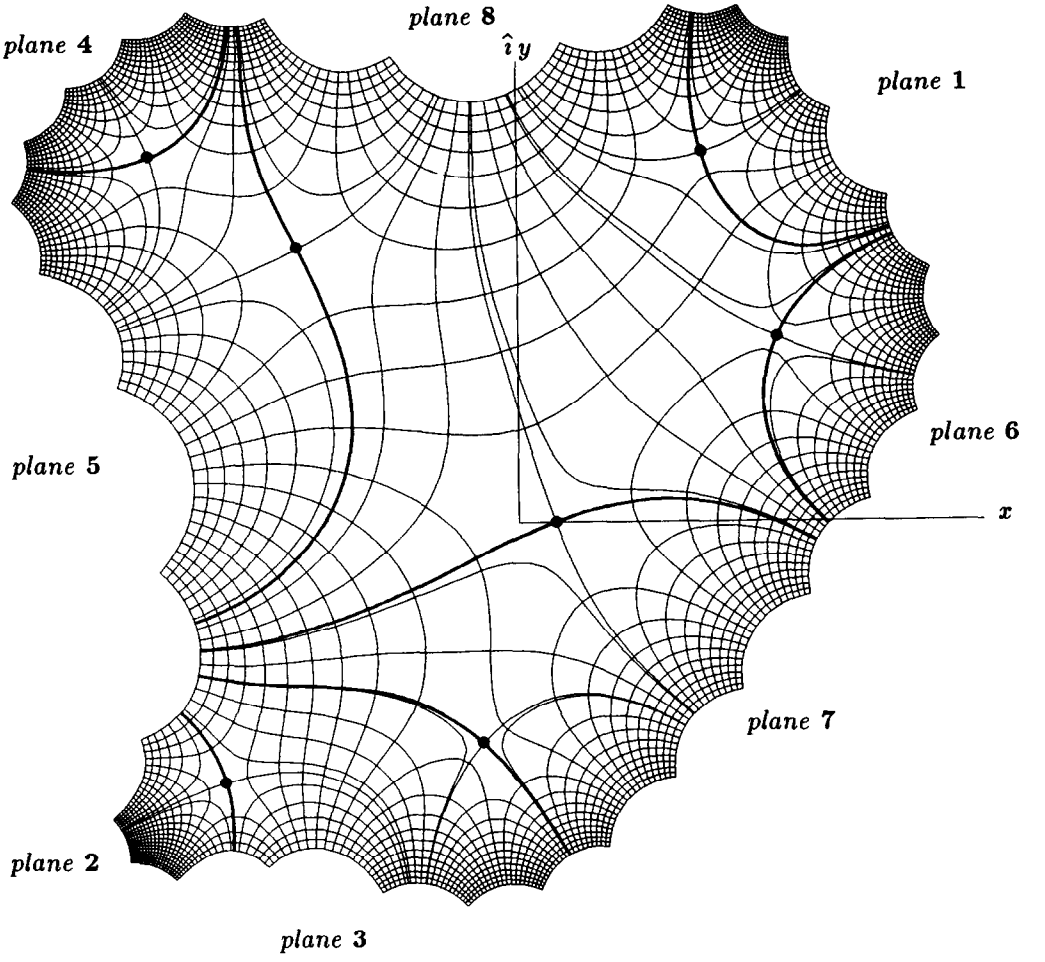
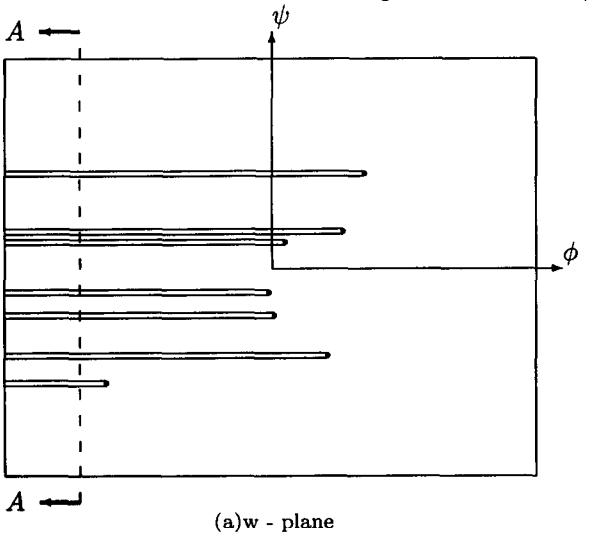
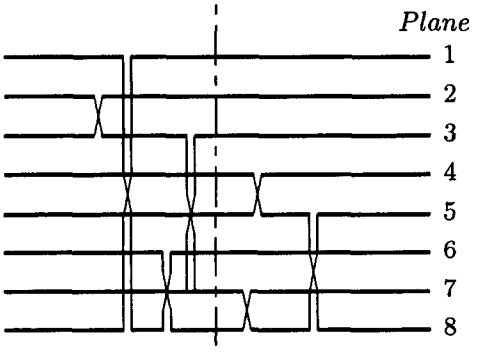


Figure 17. Function $w(z)$ in the $z-$plane.



(a)w - plane
Branches of $z(w)$.



(b)Crossection A-A.
Riemann Planes of $z(w)$.

Figure 18. 8^{th} order polynomial in the w-plane.

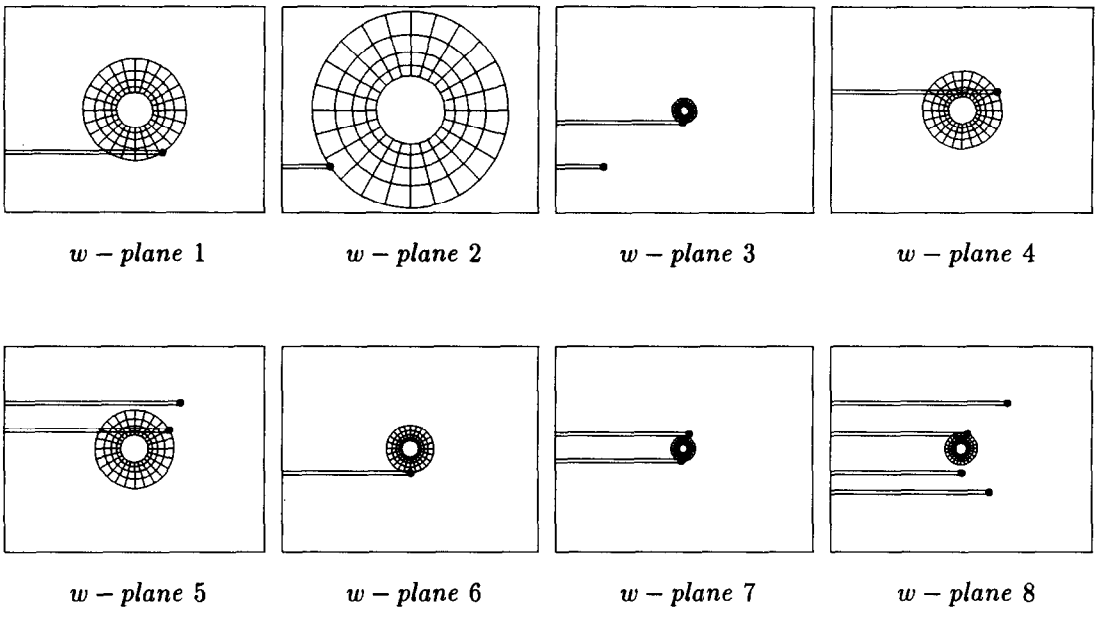


Figure 19. Convergence regions in the Riemann w-plane.

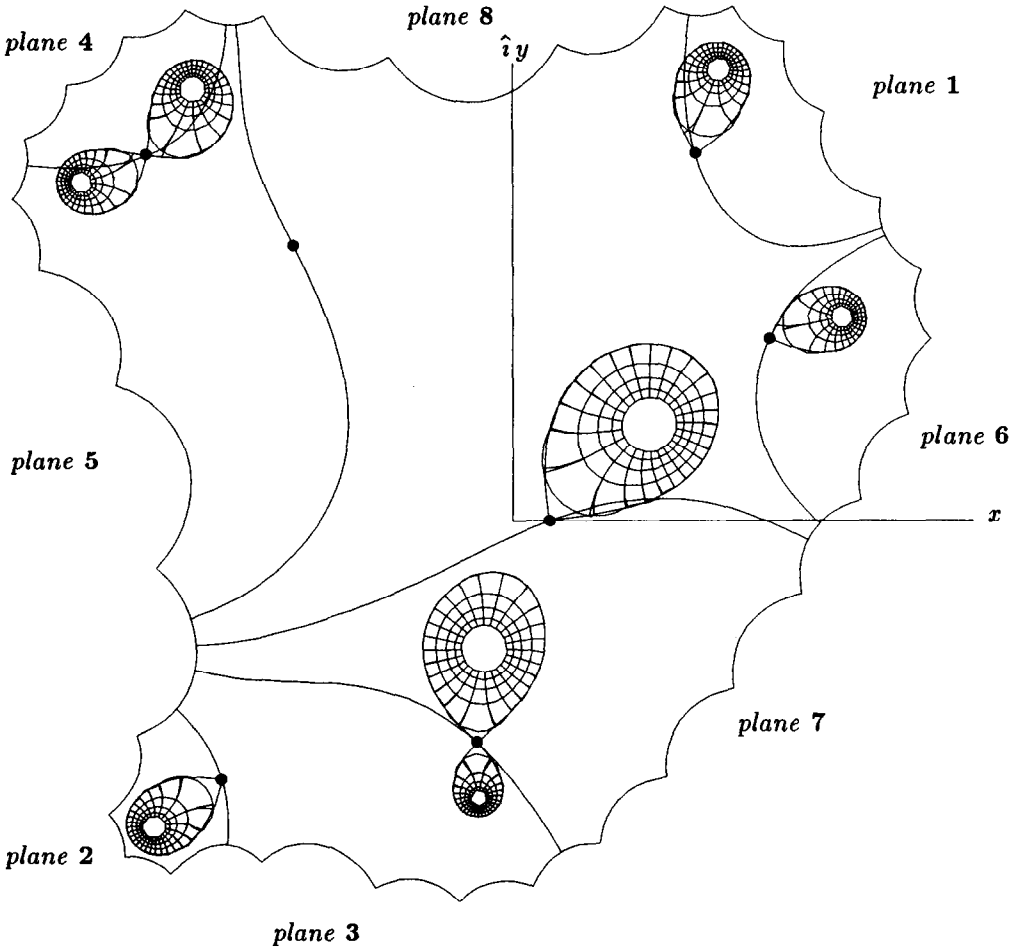


Figure 20. Convergence regions in the z-plane.

20.3. Reversion of Functions

Whichever way a function $p(u)$ is defined, the reverse is the solution of the equation $p(u(x)) = x$, which can be put in the form $f(u(x)) = p(u(x)) - x = 0$ of which the Taylor transform is $f(\overline{u}) = p(\overline{u}) - \overline{\delta}_1 = \overline{0}$. The solution by the method of Section 15 will then use the program steps for the definition of the function $p(x)$ applied to the convolution number $\overline{u}$. If the definition is in the form of a differential equation then the method of Section 17.1 can be used, which is to solve the differential equation $\overline{u}' = 1/p_u(\overline{u})$. In the program for this equation, the program for the convolution number $\overline{p}$ must be used where the integration steps must be supplemented with multiplication to transform integration w.r.t. u to integration w.r.t. x as explained in Section 18, to convert it to the program of the function $\overline{p}(\overline{u})$. This method can be considered as definition of a function, the reverse in this case, using the definition of a previous function, as explained in Section 18. As illustration consider the functions $\arcsin(x)$ and $\arccos(x)$ defined as as reverse of $\sin(u)$ and $\cos(u)$. The differential equation to be solved is

$$\overline{u}' = 1/\cos(\overline{u}) \qquad \text{for } \arcsin(x), \tag{148}$$

$$\overline{u}' = -1/\sin(\overline{u}) \qquad \text{for } \arccos(x). \tag{149}$$

Note that in the sequence of definitions given here we may not use the differential equation $u' = \dfrac{1}{\sqrt{1-x^2}}$, e.g. [12], with the simple convolution solution $\overline{u} = \int 1/\sqrt{1-\overline{\delta}^{*2}}$, if we have not proved the identity $\cos^2(x) + \sin^2(x) = 1$ from the definition equation (109). Of course, this can easily be done by multiplying the defining equation for $\sin(x)$, which is $u'' + u = 0$ by $2u'$, so that $2u''u' + 2uu' = \left(u'^2 + u^2\right)' = 0$, which together with the definition $u' = \cos(x)$ and the initial conditions prove the identity. But generally, we don't want to rely on relations other than the definition of the function. It doesn't even matter in the applications if we miss the identity of the convolution numbers from the solutions of the two different differential equations.

Returning to the solution of the differential equations (148) and (149), the functions $\cos(\overline{u})$ and $\sin(\overline{u})$ may be programmed by using the same subroutine as shown graphically in Figure 13. But we may realize that the original functions before reversion $\sin(u) = x$ and $\cos(u) = x$, respectively, can be used and merely the property $\cos(x) = \sin'(x)$ and $\sin(x) = -\cos'(x)$ of the differential equation (109) can be employed to obtain $\cos(u)$ and $\sin(u)$, respectively. The pointer programs for the convolution numbers which are the Taylor transforms of the reverse functions arcsin and arccos are then, using the classical branches of the multivalued roots u_0,

Program for $\arccos(x)$:
$u_0 = 0$
$\cos(u_0) = 1$
$-\overline{\delta}_1\{0,1,0,\ldots\}$
set virgin pointer convolution numbers:
$\overline{u}', \overline{u}, \overline{c}, \overline{c}'$
for $i = 1$ to n
$\overline{c}' = -\overline{\delta}_1 * \overline{u}'$
$\overline{c} = \int \overline{c}'dx$
$\overline{u}' = 1/\overline{c}$
next i
$\overline{u} = \int \overline{u}'$

Program for $\arcsin(x)$:
$u_0 = 0$
$\cos(u_0) = 1$
$-\overline{\delta}_1\{0,1,0,\ldots\}$
set virgin pointer convolution numbers:
$\overline{u}', \overline{u}, \overline{c}, \overline{c}'$
for $i = 1$ to n
$\overline{c}' = -\overline{\delta}_1 * \overline{u}'$
$\overline{c} = \int \overline{c}'dx$
$\overline{u}' = 1/\overline{c}$
next i
$\overline{u} = \int \overline{u}'$

These programs produce the exact coefficients according to Theorem 4. The theoretical radius of convergence is one, but the numerical radius of convergence on a IBM PC with seven digits was found for various lengths n:

n	radius
12	0.34
30	0.63
50	0.76

The analog diagrams are given in Figure 21.

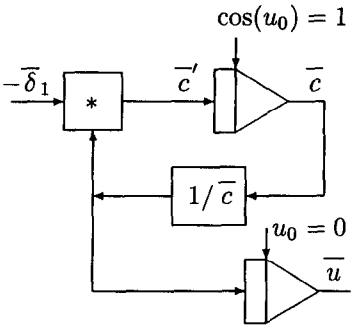


Figure 21.1. Arcsin: $\overline{u}' = 1/\overline{c}$.

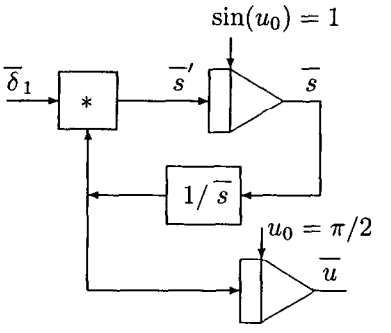


Figure 21.1. Arccos: $\overline{u}' = -1/\overline{s}$.

Again, these programs can be extended to compute $\arcsin(\overline{u})$ and $\arccos(\overline{u})$, respectively, with the transformation $f'(u) = f_u(u)u'$ as shown in Section 18.

As comparison, the same reverse functions were programmed for $n = 30$ using the Taylor series expansion for $\cos(\overline{u})$ and $\sin(\overline{u})$ in the equations (148) and (149), respectively, which is of course a clumsy program requiring the many intermediate variables. For $\arcsin(x)$ this is almost the same as the classical reversion of series and the same machine exact coefficients are obtained. But for $\arccos(x)$ this is an approximation as explained above, and the approximated coefficients differ from the exact, which are shown in Appendix 9. Yet the result is not as bad, the numerical radius of convergence has been found to be 0.61, which is only slightly worse than the 0.63 for the exact method. Nevertheless, the reversion of series is not a true convolution number operation which we have defined as a computation of an element in a finite number of steps, which, symbolically executed, leads to an analytically exact result.

Because of the periodicity of the sin and cos functions, the other branches of arcsin and arccos need not be computed as would generally be the case for the reversion of a function.

21. BI-CONVOLUTION NUMBER

The bi-convolution number is introduced for computations with bivariate Taylor expansions. It may be used for solving nonlinear equations of two variables and initial value problems of partial differential equations.

The arithmetic of bivariate polynomials is discussed in [16] and a symbolic approach to the solution of partial differential equations in [23,64]. In this treatise, the coefficients are separated from the bivariate polynomial and thus, a number defined and the properties of the number will be dicussed and its use for numerical computations demonstrated.

21.1. The Bilinear Form

The bilinear form is introduced in matrix algebra [34,54], to arrange the coefficients of two variable vectors in a $n \times m$ matrix $\underline{\overline{A}}$, counting from zero for our purposes,

$$a(x,y) \equiv Q = \underline{x} \cdot \underline{\overline{A}} \cdot \overline{y}. \tag{150}$$

The matrix need not be square or symmetric as in [54]. The bivariate polynomial is formed when the variables are defined as

$$\underline{x} \equiv [x_0, x_1, x_2, \ldots, x_n] = [1, x, x^2, \ldots, x^n], \overline{y} \equiv \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 \\ y \\ y^2 \\ \vdots \\ y^m \end{bmatrix}.$$

The subscripts on x, y are indices and superscripts are powers. If superscripts on x, y are introduced at any stage, the powers will have to be written with brackets, i.e., $(x)^2, (y)^3, (x^i)^2, (y^j)^3$.

The truncated double Taylor expansion of $a(x, y)$ can be considered a bilinear form, therefore, we use the term bilinear form, although the array is then not a matrix. We call $\underline{\overline{A}}$ the Taylor transform of $a(x, y)$. The fully notated Taylor transformation as in Section 2 is

$$\underline{\overline{A}}{}^n_m = \underline{\overline{X}}{}^n_x \cdot \underline{\overline{A}}{}^x_y \cdot \underline{\overline{Y}}{}^y_m, \tag{151}$$

where $\underline{\overline{A}}{}^x_y$ is the matrix representing $a(x, y)$ in abstract vector space, numerically approximated by sampled values of $a(x, y)$. We use $\underline{\overline{A}}$ in equation (150) as default notation for the matrix of coefficients $\underline{\overline{A}}{}^n_m$ in equation (151).

The bilinear form can be written with $\underline{\overline{A}}$ as (covariant) row vector of (contravariant) column vectors $\overline{a_i}$

$$\underline{\overline{A}} = \left[\overline{a}_0, \overline{a}_1, \overline{a}_2, \ldots \right],$$

so that

$$a(x, y) = \underline{x} \cdot \left[\overline{a}_0, \overline{a}_1, \overline{a}_2, \ldots \right] \cdot \begin{bmatrix} 1 \\ y \\ y^2 \\ \vdots \end{bmatrix},$$

$$= \underline{x} \cdot \overline{a}^0 1 + \underline{x} \cdot \overline{a}_1 y + \underline{x} \cdot \overline{a}_2 y^2 + \cdots, \tag{152}$$

or as (contravariant) column vector of (covariant) row vectors $\underline{a}^i$

$$\underline{\overline{A}} = \begin{bmatrix} \underline{a}^0 \\ \underline{a}^1 \\ \underline{a}^2 \\ \cdots \end{bmatrix},$$

so that

$$a(x, y) = [1, x, x^2 \ldots x^n] \cdot \begin{bmatrix} \underline{a}^0 \\ \underline{a}^1 \\ \underline{a}^2 \\ \cdots \end{bmatrix} \cdot \overline{y}$$

$$= 1\, \underline{a}^0 \cdot \overline{y} + x\, \underline{a}^1 \cdot \overline{y} + x^2\, \underline{a}^2 \cdot \overline{y} + \cdots. \tag{153}$$

Addition and subtraction is seen from both forms to correspond to addition and subtraction of the matrices

$$\underline{x} \cdot \overline{\underline{A}} \cdot \overline{y} \pm \underline{x} \cdot \overline{\underline{B}} \cdot \overline{y} = \pm \underline{x} \cdot \left[\overline{\underline{A}} \pm \overline{\underline{B}} \right] \cdot \overline{y}$$

Multiplication in the form of equation (152) becomes, applying Theorem 1 to each term y^j,

$$\begin{aligned} \underline{x} \cdot \overline{\underline{A}} \cdot \overline{y} \times \underline{x} \cdot \overline{\underline{B}} \cdot \overline{y} &= \underline{x} \cdot \overline{a}_0 * \overline{b}_0 \, 1 \\ &\quad + (\underline{x} \cdot \overline{a}_0 * \overline{b}_1 + \underline{x} \cdot \overline{a}_1 * \overline{b}_0) \, y \\ &\quad + (\underline{x} \cdot \overline{a}_0 * \overline{b}_2 + \underline{x} \cdot \overline{a}_1 * \overline{b}_1 + \underline{x} \cdot \overline{a}_2 * \overline{b}_0) \, y^2 \\ &\quad \vdots \\ &\equiv \underline{x} \cdot \overline{\underline{C}} \cdot \overline{y}. \end{aligned} \tag{154}$$

Multiplication in the form of equation (153) becomes, applying Theorem 1 to each term x^i,

$$\begin{aligned} \underline{x} \cdot \overline{\underline{A}} \cdot \overline{y} \times \underline{x} \cdot \overline{\underline{B}} \cdot \overline{y} &= 1 \, \underline{a}^0 * \underline{b}^0 \cdot \overline{y} \\ &\quad + x \, (\underline{a}^0 * \underline{b}_1 \cdot \overline{y} + \overline{a}^1 * \overline{b}^0 \cdot \overline{y}) \\ &\quad + x^2 \, (\underline{a}^0 * \underline{b}_2 \cdot \overline{y} + \overline{a}^1 * \overline{b}^1 \cdot \overline{y} + \underline{a}^2 * \underline{b}_0 \cdot \overline{y}) \\ &\quad \vdots \\ &\equiv \underline{x} \cdot \overline{\underline{C}} \cdot \overline{y}. \end{aligned} \tag{155}$$

The multiplications and summations in equation (154) are shown diagrammatically in Figure 22(a), analog to the convolution product diagram of Figure 1. The summation of convolution products along the diagonal connecting lines produce the convolution elements of the result, which must then be written in column form into $\overline{\underline{C}}$. The multiplications and summations in equation (155) are shown in Figure 22(b), but the resulting convolution elements must also be entered as rows into $\overline{\underline{C}}$. If now the detail of each internal convolution product is written out, it can be checked that the result of the multiplication in Figure 22(a) and in Figure 22(b) is the same matrix $\overline{\underline{C}}$.

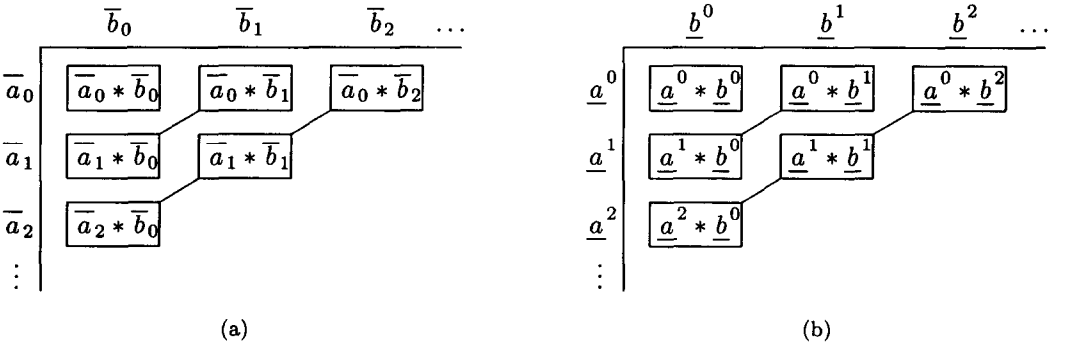


Figure 22. Multiplication of convolution number of convolution numbers.

The division operation $\underline{x} \cdot \overline{\underline{C}} \cdot \overline{y} \, / \, \underline{x} \cdot \overline{\underline{B}} \cdot \overline{y} = \underline{x} \cdot \overline{\underline{A}} \cdot \overline{y}$ can be constructed by solving either the multiplication form of equation (154) or of equation (155), analog to the convolution division in Section 3.3, where the elements are convolution numbers.

We call the matrix $\overline{\underline{A}}$ of the bilinear form a *bi-convolution number*, which is the Taylor transform of a bivariate function. The bi-convolution number may be considered to consist of

rows or of columns of convolution numbers, which we call the two *modes*, and the product of a bi-convolution number is seen as depicted in Figure 22 to consist of the convolution of convolution numbers. This leads to a formal definition of a bi-convolution number.

21.2. Definition of the Bi-Convolution Number

Consider the definition of the bi-convolution number from the perspective of an isomorphism as follows.

In the previous definition of a convolution number of Section 2, the elements were real or complex numbers. The elements could also be from another ring, e.g., matrices as in Section 4. All that is required from the elements that the four arithmetical operations on them are defined and produce another element of the ring. Since convolution numbers meet all these requirements, we may construct a convolution number with convolution numbers as elements which we call bi-convolution numbers as it will be identified as the Taylor transform of the bivariate polynomial.

Similarly, a convolution number of a convolution number of a convolution number can be defined in unlimited sequence, which are multi-convolution numbers for use with multivariate polynomials. However, this treatise is limited to the bi-convolution number.

Particularly, multiplication is commutative, because the element multiplication is commutative, and there is not a different left and right division as for matrices. This means that a covariant bi-convolution number with contravariant convolution elements is the same as a contravariant bi-convolution number with covariant convolution elements constructed from the same common matrix form. The matrix is a sort of a neutral form which allows both interpretations, and the application may decide in which order to proceed. Extensive use of the graphical arrangement of the bi-convolution number as matrix and matrix notation will be made to simplify the development.

The definition of the bi-convolution number in this subsection may not be a natural one, but its usefulness is that the definition of four arithmetic operations on the bi-convolution number allows an isomorphism with the ordinary convolution number so that without further proof we can immediately adopt all the algebra and theorems of the ordinary convolution number, with the ordinary numbers replaced by convolution numbers, in either of the two modes of the elements of the bi-convolution number, although the details may not always be so obvious. Using the isomorphism the theorems can be stated in terms of convolution elements, and then again going one level deeper, they can also be written in the detail of the number elements of the bi-convolution number. Therefore, three alternatives may be written for each theorem, each one may be useful for a particular purpose.

Leaving the development of the algebra for later, the theorems are summarized as follows: The Taylor transform of a function of two variables, called a bivariate Taylor transform, is defined by

$$u(x,y) = \underline{x} \cdot \underline{\overline{U}} \cdot \overline{y}, \tag{156}$$

where $\underline{\overline{U}}$ is the two-dimensional array of the coefficients of the two-dimensional Taylor expansion of $u(x,y)$, which is in the number domain, and $\underline{\overline{U}}$ is in the bi-convolution domain. In bi-convolution algebra, $\underline{\overline{U}}$ is a bi-convolution variable. A bi-convolution variable $\underline{\overline{F}}$ can be a scalar or bi-convolution function of another bi-convolution variable $\underline{\overline{U}}$. A convolution function of a bi-convolution variable is also possible.

THEOREM 1b. *If the Taylor transform of a function $u(x,y)$ is $\underline{\overline{U}}$ then*

$$f(u(x,y)) = \underline{x} \cdot \underline{\overline{F}}(\underline{\overline{U}}) \cdot \overline{y}, \tag{157}$$

where f is a two-dimensional functional function of u.

THEOREM 2b. *The compatibility equations of Section 10 apply to the bi-convolution number, with convolution numbers in the compatibility equations, and convolution divisions in the elements of the Jacobian matrix, in both modes.*

From this theorem follows the same theory of solution of nonlinear and partial differential equations by recurrence schemes, and its implementation by a pointer bi-convolution number. They are stated here for the regular bi-convolution number and are modified for the singular bi-convolution number according to the same rules as in Section 14.1.

THEOREM 3b. *The leading convolution element $\overline{f}_0$ of a bi-convolution function $\underline{\overline{F}}(\underline{\overline{U}})$ contains only the leading column $\overline{u}_0$, and the function $\underline{x} \cdot \overline{f}_0(\overline{u}_0) = f(u(x,0),0)$.*

Applying Theorem 3 again to this statement, we obtain the following theorem.

THEOREM 3c. *The leading element F_{00} of a convolution function $\underline{\overline{F}}(\underline{\overline{U}})$ contains only the leading element U_{00}, and the function $F_{00}(U_{00}) = f(u(0,0),0)$. In the solution of a nonlinear equation $f(\underline{\overline{U}}) = \underline{\overline{0}}$ the leading element U_{00} has to be determined nonlinearly from the scalar equation $f(U_{00}) = 0$.*

THEOREM 4b. *The convolution elements $\overline{f}_i$ are functions of the convolution elements $\overline{u}_{k<i}$ only.*

THEOREM 4c. *The elements F_{ij} are functions of the elements $U_{k<i,l<j}$ only.*

THEOREM 5b. *A bi-convolution polynomial of degree n has at most n bi-convolution roots.*

A bi-convolution differential exists and is the Taylor transform of the variation of a function of two variables in the sense of Variational Calculus.

A bi-convolution derivative $d\underline{\overline{F}} \not\downarrow d\underline{\overline{U}}$ exists.

THEOREM 6b. *If the Taylor transform of a function $u(x,y)$ is $\underline{\overline{U}}$ then*

$$L(u(x,y)) = \underline{x} \cdot \underline{\overline{L}}(\underline{\overline{U}}) \cdot \overline{y} \tag{158}$$

where L is a two-dimensional scalar, functional, partial differential and partial integro operator, and $\underline{\overline{L}}(\underline{\overline{U}})$ is the Taylor transform of $L(u(x,y))$.

21.3. Arithmetic of the Bi-Convolution Number

The computer storage of the bi-convolution number is shown in Figure 23, in column mode according to equation (154) in (a) and in row mode according to equation (155) in (b).

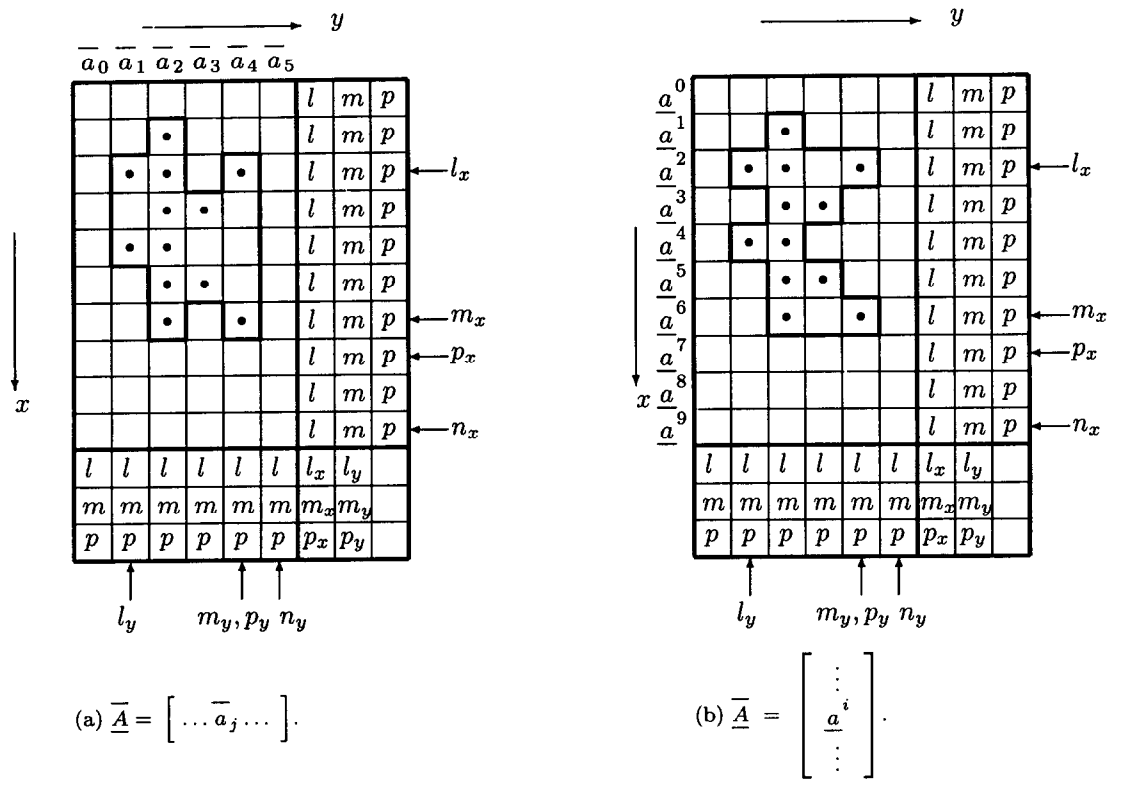


Figure 23. Bi-convolution number storage.

For convenience, we refer to the two dimensions as x and y as shown, according to the form of equation (150). Each convolution element in both modes has its internal pointers l and m stored individually at the end. The convolution element pointers in either direction l_x, l_y, m_x, m_y refer to the lowest and highest convolution number in the 2 directions respectively, which we call global pointers. Also stored are the individual pointers p of each column and each row convolution element. The global pointers p_x, p_y are the maximum of all pointers p in the x and y directions, respectively, but these are only necessary and not sufficient and do not describe the state as completely as in an ordinary convolution number. A pointer bi-convolution number is a pointer convolution number whose elements consist of pointer convolution numbers in both modes, but here the isomorphism breaks down because convolution elements are not only completely determined or completely undetermined which will be clear from the examples. If there are two bi-convolution numbers $\underline{\overline{A}}$ and $\underline{\overline{B}}$ then we refer to the corresponding pointers as l_{ax}, l_{bx}, m_{ax}, m_{by}, p_{ax}, p_{ay}, respectively.

The bi-convolution arithmetic operations are obtained by rewriting the convolution operations with number elements replaced by convolution elements in either of the two modes.

$$\text{If} \quad \underline{\overline{A}} \pm \underline{\overline{B}} = \underline{\overline{C}}, \quad \text{then} \quad \overline{c}_j = \overline{a}_j \pm \overline{b}_j \quad \text{or} \quad \underline{c}^i = \underline{a}^i \pm \underline{b}^i. \tag{159}$$

Note that the symbols as well as the operations are indistinguishable from the corresponding matrix operations.

$$\text{If} \quad \underline{\overline{A}} * \underline{\overline{B}} = \underline{\overline{C}} \quad \text{then} \quad \overline{c}_j = \sum_{k=0}^{n} \overline{a}_k * \overline{b}_{j-k} \quad \text{or} \quad \underline{c}^i = \sum_{k=0}^{n} \underline{a}^k * \underline{b}^{i-k}. \tag{160}$$

Here the convolution multiplication symbol $*$ distinguishes this operation from a matrix $\cdot$ multiplication. Similarly we denote convolution divsion by the symbol $\not*$ to distiguish it from right

matrix division. The division formula is, analog to the division in Section 3.3,

$$\text{If } \underline{\overline{C}} \not{*} \underline{\overline{B}} = \underline{\overline{A}}, \quad \text{then } \overline{a}_j = \left(\overline{c}_{j+l_{by}} - \sum_{k=l_{by}+1}^{n} \overline{a}_{j+l_{by}-k} * \overline{b}_k \right) / \overline{b}_{l_{by}} \tag{161}$$

$$\text{or } \underline{a}^i = \left(\underline{c}^{i+l_{by}} - \sum_{k=l_{bx}+1}^{n} \underline{a}^{i+l_{bx}-k} * \underline{b}^k \right) / \underline{b}_{l_{bx}} \tag{162}$$

where the limit pointers l_{bx} and l_{bx} are described below to allow compatible division of bi-convolution numbers with zero leading rows or columns.

For pointer bi-convolution numbers all the summation formulas in equations (160) and (161) or (162) must be programmed specially to avoid an unspecified large number of intermediate variables that are used in ordinary pointer convolution numbers addition.

Note that a bi-convolution number doesn't have to be square to enable division, in fact, the ordinary convolution number can be considered as a single column bi-convolution number. Also, while the identity matrix is the square array

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{bmatrix},$$

the identity bi-convolution number is a rectangular array

$$\begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots \\ \vdots & & & \end{bmatrix}.$$

The square root routine of Section 3.4 is adapted for the bi-convolution number:

$$\text{If } \sqrt{\underline{\overline{C}}}^{\,*} = \underline{\overline{A}}, \quad \text{then in column mode}$$

$$\begin{aligned} l_{ay} &= l_{cy}/2 \\ \overline{a}_{l_{ay}} &= \sqrt{\overline{c}_{l_{cy}}} \\ \overline{a}_i &= \left(\overline{c}_{l_{ay}+i} - \sum_{j=l_{ay}+1}^{i-1} \overline{a}_j * \overline{a}_{i+l_{ay}-j} \right) / 2\overline{a}_{l_{ay}}, \end{aligned} \tag{163}$$

and similar for row mode. The formula should be programmed taking advantage of the symmetry. The bi-convolution square root symbol distinguishes it from the square root of a square matrix. The bi-convolution number $\underline{\overline{C}}$ need not be square to have a square root.

The general power $\underline{\overline{A}}^{*k}$, where k is any real number, is programmed as a function defined by a partial differential equation.

A matrix and a bi-convolution number have the same array structure of numbers, graphically displayed as a rectangle. The distinction between the two is not the structure but the use, leading to a different algebra. Therefore, a bi-convolution number cannot be defined as a rectangular array of elements, an error that is made in many texts in the definition of a matrix, e.g., [54], instead of deriving it from its application, [34].

Polynomials in $x, 1/y$, $1/x, y$ and $1/x, 1/y$ are transformed to bivariate polynomials with positive powers, similar to the transformation in Example 2 of Section 2, to apply bi-convolution algebra.

Null operations may occur in various convolution elements but not all. The null operations for the operations on all columns or rows must be accumulated to generate a total null operation.

As shown in Figure 23, a typical distribution of nonzero elements results in different boundaries in the x and y directions. Closely connected to this is the fact that it is not possible to see from the element or total boundaries whether a division is compatible. Whichever method of Figure 23(a) or (b) is followed, the convolution limits l_x or l_y must be compatible, then the division according to equations (161) or (162) can be started, but in this routine every division step $/\,\overline{b}_j$ or $/\,\underline{b}^i$ must be compatible. Depending on which mode is chosen, any incompatibility may show up by the convolution limits l_x or l_y, or these may be compatible and the incompatibility may show up in one of the steps of the division routine.

As an example, consider the division

$$\frac{x+xy+y^2}{x+y}=\frac{\underline{x}\cdot\overline{\underline{C}}\cdot\overline{y}}{\underline{x}\cdot\overline{\underline{B}}\cdot\overline{y}}.$$

The global limits l_x and l_y of both $\overline{\underline{C}}$ and $\overline{\underline{B}}$ are 0 and the first division in equation (161), $\overline{c}_0/\,\overline{b}_0$, or the first division in equation (162), $\overline{c}_0/\,\overline{b}_0$, can be executed, but the next division will be found to be incompatible.

On the other hand, consider the division

$$\frac{1+x+y}{y+xy}=\frac{\underline{x}\cdot\overline{\underline{C}}\cdot\overline{y}}{\underline{x}\cdot\overline{\underline{B}}\cdot\overline{y}}.$$

The method by columns, equation (161), immediately shows that division is incompatible because $l_y(C) < l_y(B)$. But the method of rows, equation (162), will start because $l_x(C) = l_x(B) = 0$ and incompatibility will only show up at the first division $\underline{c}^0/\,\underline{b}^0$. As the next example, take the division

$$\frac{xy}{x+y}=\frac{\underline{x}\cdot\overline{\underline{C}}\cdot\overline{y}}{\underline{x}\cdot\overline{\underline{B}}\cdot\overline{y}}.$$

Each column of $\overline{\underline{C}}$ is divisible by each column of $\overline{\underline{B}}$ and each row of $\overline{\underline{C}}$ is divisible by each row of $\overline{\underline{B}}$. The global limits l_x and l_y are both compatible and division by either formula will proceed and the incompatibility only found at the step when $\overline{a}_2$ or $\underline{a}^2$ has to be determined.

Let us now follow the steps of the simple division

$$\frac{1}{x+k+y}=\frac{\underline{x}\cdot\overline{\underline{C}}\cdot\overline{y}}{\underline{x}\cdot\overline{\underline{B}}\cdot\overline{y}}=\underline{x}\cdot\overline{\underline{A}}\cdot\overline{y}.$$

Here,

$$\overline{\underline{C}}=\begin{bmatrix}1&0&0&\cdots\\0&&&\\0&&&\\\vdots&&&\end{bmatrix},\quad \overline{\underline{B}}=\begin{bmatrix}k&1&0&\cdots\\1&&&\\0&&&\\\vdots&&&\end{bmatrix}.$$

Therefore, we have to compute $\overline{\underline{A}} = \overline{\underline{C}} \not{*} \overline{\underline{B}}$ by solving the equation $\overline{\underline{A}} * \overline{\underline{B}} = \overline{\underline{C}}$. From the method of equation (162), Figure 23(a),

$$\overline{a}_0 * \overline{b}_0 = \overline{c}_0 = \overline{\delta}_0,$$

therefore,

$$\overline{a}_0 = \overline{\delta}_0 / \overline{b}_0 = 1/ \begin{bmatrix} k \\ 1 \end{bmatrix} = \begin{bmatrix} 1/k \\ -1/k^2 \\ 1/k^3 \\ -1/k^4 \\ \vdots \end{bmatrix}.$$

The next diagonal is

$$\overline{a}_1 * \overline{b}_0 + \overline{a}_0 * \overline{b}_1 = \overline{c}_1 = \overline{0},$$

therefore,

$$\overline{a}_1 = -\overline{a}_0 * \overline{b}_1 / \overline{b}_0 = -\overline{a}_0 \cdot 1/\overline{b}_0 = - \begin{bmatrix} 1/k \\ -1/k^2 \\ 1/k^3 \\ -1/k^4 \\ \vdots \end{bmatrix} / \begin{bmatrix} k \\ 1 \end{bmatrix} = \begin{bmatrix} -1/k^2 \\ 2/k^3 \\ -3/k^4 \\ \vdots \end{bmatrix}.$$

Continuing, the result is

$$\underline{\overline{A}} = \begin{bmatrix} 1/k & -1/k^2 & 1/k^3 & -1/k^4 & \cdots \\ -1/k^2 & 2/k^3 & -3/k^4 & \cdots & \\ 1/k^3 & -3/k^3 & \cdots & & \\ -1/k^4 & \cdots & & & \\ \vdots & & & & \end{bmatrix}.$$

This formula can be used to check a programmed division routine.

Without proof, we assume that pointer limits of the different convolution elements may differ, so that the pointer limit boundary has an echelon form, but that this echelon form can only be convex to preserve the dual column/row mode interpretation of elements. Therefore, it can also not be determined from the global pointers p_x and p_y alone, whether a pointer convolution operation is a null operation or overdetermined.

The radius of convergence applies for both the complex x-region for every value of y, as well as the complex y-region for every value of x.

21.4. Partial Differentiation and Integration

Differentiation of the two-dimensional function consists of partial differentiation $\frac{\partial}{dx}$ and $\frac{\partial}{dy}$. The operation $\frac{\partial}{dx}$ is applied to each column as the operation $\frac{d}{dx}$ in Section 12 with the pointer shift in each column according to Section 14. Similarly, the operation $\frac{\partial}{dy}$ is applied to each row. The operations can also be programmed on rows and columns, respectively, as for single convolution numbers. In any case, a subroutine library should be either completely in column mode or completely in row mode.

Integration is replaced by "partial" integration, which is written as $\int u(x,y)\,dx$ and $\int u(x,y)\,dx$. The integration routines are applied to columns for the x-integration and to rows for the y-integration according to equation (86) in Section 12 with pointer shift according to Section 14. Note that replacing the numbers in equations (85), (86) by convolution elements may not be an efficient programming method because each column has its individual pointers.

It is suggested that in the integration routine the complete convolution constant initial value, with pointer, is supplied to remind the programmer and to have an appropriate place in the program where to apply the initial value function. That is, in

$$\int \underline{x} \cdot \underline{\overline{A}} \cdot \overline{y}\, dx = \underline{x} \cdot \underline{\overline{B}} \cdot \overline{y}$$

the convolution number $\overline{b}_0$ of the Taylor series $\underline{x} \cdot \overline{b}_0$ of the initial function $b(0,y)$ must be supplied.

The matrix representation of the bi-convolution number allows us to use the same differentiation and integration symbols of Section 12,

$$\int u(x,y)\, dx = u^I(x,y) = \underline{x} \cdot \underline{\overline{U}}^I \cdot \overline{y}$$

$$\int u(x,y)\, dy = u_I(x,y) = \underline{x} \cdot \underline{\overline{U}}_I \cdot \overline{y}$$

$$\frac{\partial u(x,y)}{dx} = u'(x,y) = \underline{x} \cdot \underline{\overline{U}}' \cdot \overline{y}$$

$$\frac{\partial u(x,y)}{dy} = u_{,}(x,y) = \underline{x} \cdot \underline{\overline{U}}_{,} \cdot \overline{y}$$

which in the bi-convolution domain is independent of the variable names. Alternatively,

$$\int u(x,y)\, dx = u^{(X)}(x,y) = \underline{x} \cdot \underline{\overline{U}}^{(X)} \cdot \overline{y}$$

$$\int u(x,y)\, dy = u_{(Y)}(x,y) = \underline{x} \cdot \underline{\overline{U}}_{(Y)} \cdot \overline{y}$$

$$\frac{\partial u(x,y)}{dx} = u^{(x)}(x,y) = \underline{x} \cdot \underline{\overline{U}}^{(x)} \cdot \overline{y}$$

$$\frac{\partial u(x,y)}{dy} = u_{(y)}(x,y) = \underline{x} \cdot \underline{\overline{U}}_{(y)} \cdot \overline{y}$$

$$\frac{\partial u(x,y)}{dy} = u_{(y)}(x,y) = \underline{x} \cdot \underline{\overline{U}}_{(y)} \cdot \overline{y}$$

which can also be used for multivariate functions and is still distinct from the custumary component notation. Having provided for distinctive notation if necessary, we can still use the customary partial differential notation u_x, u_y, u_{xy}, and their transforms $\underline{\overline{U}}_x$, $\underline{\overline{U}}_y$, $\underline{\overline{U}}_{xy}$ if the meaning is clear from the context. The integrals are then denoted by u_X, u_Y, u_{XY}, and their tranforms by $\underline{\overline{U}}_X$, $\underline{\overline{U}}_Y$, $\underline{\overline{U}}_{XY}$. Because both aspects are equally important in convolution notation, the convolution equation should look as similar as possible as the equation in the number domain, and the symbols should identify convolution variables and operations uniquely and distinguish them from number domain variables and operations.

The functions generated from their defining differential equations in Section 18 can be generated for the bi-convolution number in the same way, where integration is replaced by partial integration w.r.t. either the x or y variable. As an example, the general power $\underline{\overline{U}}^{*\alpha}$, where α is any real number can be programmed as shown in the middle loop of Figure 13 in Section 18, adapted for the bi-convolution number. The derivation of the partial differential equation is, using the x variable,

$$\underline{\overline{F}} \equiv \underline{\overline{U}}^{*\alpha},$$

$$d\underline{\overline{F}} \not* d\underline{\overline{U}} = \alpha \underline{\overline{U}}^{*\alpha-1} = \alpha \underline{\overline{U}}^{*\alpha} \not* \underline{\overline{U}} = \alpha \underline{\overline{F}} \not* \underline{\overline{U}},$$

$$\frac{\partial \underline{\overline{F}}}{dx} \equiv \underline{\overline{F}}' = d\underline{\overline{F}} / d\underline{\overline{U}} * \underline{\overline{U}}' = \alpha \underline{\overline{F}} * \underline{\overline{U}}' \not* \underline{\overline{U}}.$$

The integration analog flow diagram is exactly the same as the one in Figure 13, but the initial value to be supplied to the integrator is $\overline{F}_0 = \overline{U}_0^{*\alpha}$ which is the convolution number exponential function applied to the 0-column element of the bi-convolution variable $\underline{\overline{U}}$, executed with the routine in Figure 13 on an ordinary convolution number, with convolution numbers $\overline{U}$ and $\overline{U}_0'$ and initial value U_{00} supplied to that integrator. The total function routine therefore consists of the computation on the convolution number $\overline{U}_0$ according to the diagram in Figure 13, followed by the computation on the bi-convolution number $\underline{\overline{U}}$ also according to Figure 13, which is shown with bi-convolution notation in Figure 24. In this routine, a complete column convolution number $\overline{U}_j$ is computed in each loop.

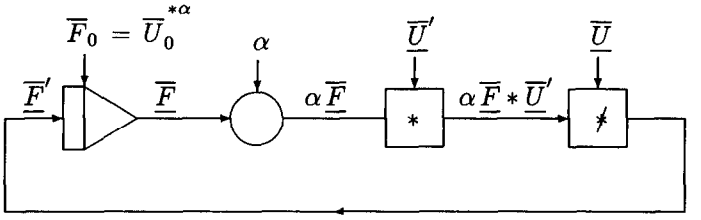


Figure 24. Integration of $\underline{\overline{F}}' = \alpha \underline{\overline{F}} * \underline{\overline{U}}' \not* \underline{\overline{U}} = \frac{\partial}{dx} \underline{\overline{U}}^{*\alpha}$.

21.5. Initial Value Problem

EXAMPLE 5. As an application of bi-convolution analysis we attempt the solution of the nonlinear 2nd order partial differential equation of the axi-symmetric gas dynamic flow at the throat of a nozzle, posed as initial value problem.

A good description of the problem can be found in [65,66]. Earlier attempts were made by using the approximate gas dynamic perturbation equation about sonic flow, called the transonic equation

$$\phi_{yy} + \frac{\phi}{y} = (\gamma + 1)\phi_x \phi_{xx}, \tag{164}$$

where the $x-$axis is the flow axis of the nozzle. Methods of power series solution have been used early [67,68,69], but with such few terms that no question of convergence arises. Since the digital computer was not yet invented, no comparison with other computational results could be made. Later, the method of Sauer, [70], also described in [65,66], became popular. The solution is obtained by solving the problem for the simple initial value $\phi_x = \alpha x$ by power series, which leads to the exact solution of the approximate equation (164)

$$\phi = \alpha x^2/2 + \phi_{12}\, xy^2 + \phi_{04}\, y^4. \tag{165}$$

The streamlines away from the centerline have a nozzle throat like shape. Another approximate solution by [71], also described in [65], is actually a boundary value method, only the distinction is not so clear because so few terms were used. Still later, the full gas-dynamic equation was solved for this problem analytically [72,73], but by a perturbation method. The full gas-dynamic dimensionless equation is [74], with the mean flow at Mach number 1,

$$\left(-\frac{\gamma+1}{2}+\frac{\gamma+1}{2}\phi_x^2+\frac{\gamma-1}{2}\phi_y^2\right)\phi_{xx}+\left(-\frac{\gamma+1}{2}+\frac{\gamma-1}{2}\phi_x^2+\frac{\gamma+1}{2}\phi_y^2\right)\phi_{yy}$$
$$+2\phi_x\phi_y\phi_{xy}-\left(\frac{\gamma+1}{2}-\left(\frac{\gamma-1}{2}\phi_x^2+\frac{\gamma-1}{2}\phi_y^2\right)\right)\frac{\phi_y}{y}=0.$$

For comparison with previous results we rather consider the corresponding exact equation for the perturbation potential

$$\left(1-\left((\gamma-1)\phi_x+\frac{\gamma-1}{2}\phi_x^2+\frac{\gamma+1}{2}\phi_y^2\right)\right)\phi_{yy}-\left((\gamma+1)\phi_x+\frac{\gamma+1}{2}\phi_x^2+\frac{\gamma-1}{2}\phi_y^2\right)\phi_{xx}$$
$$-2(1+\phi_x)\phi_y\phi_{xy}-\left(1-\left((\gamma-1)\phi_x+\frac{\gamma-1}{2}\phi_x^2+\frac{\gamma-1}{2}\phi_y^2\right)\right)\frac{\phi_y}{y}=0 \qquad (166)$$

which is conveniently written as

$$B\phi_{yy}-A\phi_{xx}-C\phi_{xy}+D\frac{\phi_y}{y}=0. \qquad (167)$$

The initial value on the axis must be chosen so that the streamlines away from the axis are curved like a nozzle throat. As a design problem the initial value problem is much better suited than the boundary value problem. Actually, the initial value problem is not applicable in the classic sense because equation (167) is not parabolic. It is elliptic in the subsonic region, for which the Cauchy problem is ill posed. Yet a solution is not impossible. On the supersonic side, equation (167) is hyperbolic and the initial value problem can be solved because initial line on the axis cross the characteristics. In the application of the flow through a nozzle, it is useful enough to obtain a solution on the supersonic side only in the region between the sonic line and the rearward characteristic from the sonic line at the axis to the wall. The remainder of the supersonic part can be designed by the method of characteristics.

However, another mathematical difficulty lies elsewhere. Even if a solution in a fairly large subsonic region is obtained, its analytical continuation backwards doesn't lead to the uniform source flow that is given in the practical application. Since this is again a Cauchy problem, we can only surmise that in future a practical approximation can be found by a different approach, onethat matches a far uniform flow to the flow in the throat in some cases and therefore an exact solution of the throat region is useful. However, it will definitely not be possible for all cases.

Generally, a perturbation equation is an approximation, but here we are interested in the exact solution. An exact perturbation equation is only possible because the original equation contains positive integer powers only. This is not the case with the equation of the stream function ψ, [74], where the density computed in terms of ψ contains a power of the real number γ. With the method of bi-convolution numbers the exact streamfunction could be treated just as well, with just a very few more algebraic steps.

The develop the solution routine, the Taylor transform of equation (167) is written in bi-convolution numbers with intermediate bi-convolution variables as main coefficients

$$\underline{\overline{B}}*\underline{\overline{\phi}}_{yy}-\underline{\overline{A}}*\underline{\overline{\phi}}_{xx}-\underline{\overline{C}}*\underline{\overline{\phi}}_{xy}+\underline{\overline{D}}*\underline{\overline{\phi}}_{y}\not{*}\underline{\overline{Y}}=\underline{\overline{0}}, \qquad (168)$$

where the bi-convolution coefficients have to be computed in bi-convolution arithmetic from $\underline{\overline{\phi}}_x$ and $\underline{\overline{\phi}}_y$, and the Taylor transform of y is

$$\underline{\overline{Y}}=\begin{bmatrix}0 & 1 & 0 & \cdots\\ 0 & & & \\ \vdots & & & \end{bmatrix}.$$

The initial value is given on the x-axis as function of x, $\phi_x(x,0) = u(x)$ which, in convolution numbers, becomes $\overline{\phi}_x = \overline{u}$. Here, $u(x,y)$ is the x-component of the perturbation velocity, the x-component of the total velocity being $1+u(x,y)$. Having the initial value given on the x-axis, the integration must proceed in the direction of the y-axis. Therefore, the highest partial derivative $\frac{\partial}{dy}$ is isolated and the solution routine starts with the integration of that term. We notice however that the division ϕ_y/y reduces any power y^k in ϕ_y by the same amount than the partial derivative $\frac{\partial \phi_y}{dy}$. In terms of bi-convolution numbers, the partial derivative $\frac{\partial}{dy}$ moves one column of $\underline{\overline{\phi}}_y$ and the pointer back by the same amount as the division $\underline{\overline{\phi}}_y \not\ast \underline{\overline{Y}}$, *viz.* one position. Both terms must therefore be subjected together by one operator L by the same method as described in Section 17.3 for the ordinary convolution number and are therefore grouped together. Arrange the terms of equation (168)

$$\begin{aligned}\underline{\overline{\phi}}_{yy} + \underline{\overline{D}} \not\ast \underline{\overline{B}} * \underline{\overline{\phi}}_y \not\ast \underline{\overline{Y}} &= \underline{\overline{A}} \not\ast \underline{\overline{B}} * \underline{\overline{\phi}}_{xx} + \underline{\overline{C}} \not\ast \underline{\overline{B}} * \underline{\overline{\phi}}_{xy} \\ &= \underline{\overline{\phi}}_{yy} + \underline{\overline{E}} * \underline{\overline{\phi}}_y \not\ast \underline{\overline{Y}} = \underline{\overline{F}} * \underline{\overline{\phi}}_{xx} + \underline{\overline{G}} * \underline{\overline{\phi}}_{xy} \end{aligned} \tag{169}$$

Converting the method of Section 17.3 from scalar to convolution elements, the first column $\overline{e}_0$ of the bi-convolution coefficient $\underline{\overline{E}}$ is extracted and substituted by $\overline{0}$,

$$\begin{aligned}\underline{\overline{E}} &= \left[\begin{array}{c:c} \overline{e}_0 & \underline{\overline{0}} \end{array}\right] + \left[\begin{array}{c:c} \overline{0} & \underline{\overline{R}}_1 \end{array}\right] \\ &= \left[\begin{array}{c:c} \overline{e}_0 & \underline{\overline{0}} \end{array}\right] + \underline{\overline{E}}_1 .\end{aligned}$$

Furthermore, compute $\underline{\overline{E}}_1 \not\ast \underline{\overline{Y}} = \underline{\overline{P}}$ so that equation (169) becomes

$$\underline{\overline{\phi}}_{yy} + \overline{e}_0 * \underline{\overline{\phi}}_y \not\ast \underline{\overline{Y}} = \underline{\overline{F}} * \underline{\overline{\phi}}_{xx} + \underline{\overline{G}} * \underline{\overline{\phi}}_{xy} - \underline{\overline{P}} * \underline{\overline{\phi}}_y . \tag{170}$$

The operation $\overline{e}_0 * \underline{\overline{\phi}}_y$ is the equivalent of a scalar $\times$ convolution number and means that $\overline{e}_0$ must be multiplied with every column of $\underline{\overline{\phi}}$, which is the same as $\left[\begin{array}{c:c} \overline{e}_0 & \underline{\overline{0}} \end{array}\right] * \underline{\overline{\phi}}$. This operation is not computed, instead the terms on the right constitute a new variable

$$\underline{\overline{\phi}}_{yy} + \overline{e}_0 * \underline{\overline{\phi}}_y \not\ast \underline{\overline{Y}} \equiv \underline{\overline{\phi}}_{ly}, \tag{171}$$

and an operator must be programmed to produce $\underline{\overline{\phi}}_y$. To find the operator means to find the inverse of an operation developed according to the equations

$$V(x,y) = \left(\frac{\partial}{dy} + e(x)\right) U(x,y),$$

$$\left(\frac{\partial}{dy} + \underline{x} \cdot \overline{e}\right) \underline{x} \cdot \underline{\overline{V}} \cdot \overline{y} = \underline{x} \cdot \underline{\overline{U}} \cdot \overline{y},$$

and considering one power of y, i.e., one column of $\underline{\overline{U}}$

$$\begin{aligned}\underline{x} \cdot \overline{v}_{k-1} y^{k-1} &= \left(\frac{\partial}{dy} + \underline{x} \cdot \overline{e}\right) \underline{x} \cdot \overline{u}_k y^k \\ &= \underline{x} \cdot \left(\frac{\partial}{dy} \overline{u}_k y^k + \overline{e} * \overline{u}_k \frac{y^k}{y}\right) \\ &= \underline{x} \cdot \left(k \overline{u}_k + \overline{e} * \overline{u}_k\right) y^{k-1},\end{aligned}$$

therefore,

$$\overline{u}_k = \overline{v}_{k-1}/(k+\overline{e}). \tag{172}$$

This is the convolution operation to be applied to each column of the bi-convolution number, corresponding to the scalar operation in equation (115) of Section 17.3 which had to be applied to each element of the convolution number. At the same time, it is now clear how the operation $\overline{e} * \underline{\overline{U}}$ must be interpreted, which is the isomorphism of the scalar operation $e\overline{u}$ of Section 17.3. The equation to be programmed is now, introducing some new intermediate variables,

$$\begin{aligned} \underline{\overline{\phi}}_{yy} + \overline{e}_0 * \underline{\overline{\phi}}_y \not\neq \underline{\overline{Y}} &= \underline{\overline{F}} * \underline{\overline{\phi}}_{xx} + \underline{\overline{G}} * \underline{\overline{\phi}}_{xy} - \underline{\overline{P}} * \underline{\overline{\phi}}_y \\ &= \underline{\overline{F}}\phi + \underline{\overline{G}}\phi - \underline{\overline{P}}\phi \\ &= \underline{\overline{F}}\phi + \underline{\overline{H}}\phi \\ &= \underline{\overline{\phi}}_{ly} \end{aligned} \tag{173}$$

The main analog diagram according to equation (170) is shown in Figure 25. The special operator is shown by a widened integrator symbol. All operations in the y-direction are shown horizontally and all operations in the x-direction vertically down, like the directions in the bi-convolution number storage. To obtain the initial convolution value $\overline{\phi}_0$ for the integrator $\int dx$, an ordinary convolution integration of the initial convolution value $\overline{\phi}_{x0}$ must be done. The second initial convolution value for the second order partial differential equation is the vertical velocity component on the axis $v(x,0) \rightarrow \overline{\phi}_{y0}$, which is the initial convolution for the first operator. This is not really necessary since in the axially symmetric flow this must be $\overline{0}$ and any other value will not be accepted. But this scheme allows to be converted to the plane flow case where the operator becomes the first integrator. The value $\overline{0}$ ensures that the division $\underline{\overline{\phi}} \not\neq \underline{\overline{Y}}$ is compatible. Even though this operation is not done during the solution, it will be done when the function is evaluated on the x-axis.

The initial condition dictates that integration must take place in the y-direction, therefore in the x-direction differentiation must be done, the differentiators indicated in the Figure 25 by another special symbol. Due to this differentiation, some of the length of the bi-convolution number in the x-direction is lost. To compute $\underline{\overline{\phi}}_{xy}$, the differentiation $\frac{\partial}{dx}(\underline{\overline{\phi}}_y)$ is made, not $\frac{\partial}{dy}(\underline{\overline{\phi}}_x)$ otherwise another column in the y-direction would be lost. This choice is not necessary for a correct solution though, in fact it is an advantage that given a few choices each will work.

The development of the bi-convolution variables with a size of $n{\times}m = 5{\times}7$, for the initial condition $u(x,0) = 0.5x$ is shown diagrammatically in Appendix 10, the bi-convolution arithmetic routines being programmed in column mode. All computations were done on a IBM PC with seven digit precision.

The same diagram also shows the definition of the intermediate bi-convolution variables. For the purpose of illustration, no programming simplification is made, e.g., storing $\underline{\overline{B}}$ and $\underline{\overline{B}}1$, $\underline{\overline{D}}$ and $\underline{\overline{D}}1$, $\underline{\overline{E}}$ and $\underline{\overline{E}}1$ in the same number, respectively. We suggest that such simplifications waste more starting time value then they save computer storage value. Once the program produces correct answers, many more simplifications can be implemented.

The addition of new computed columns for each loop is shown in thick outline, within the grid representing the $5{\times}7$ positions (counted from 0) of the bi-convolution number. Positions with zeros are empty and positions with nonzero values are indicated by a bullet •. The first column of $\underline{\overline{\phi}}_y$, and, therefore, the second column of $\underline{\overline{\phi}}$, must be zero, and for this reason two new columns are developed in each loop.

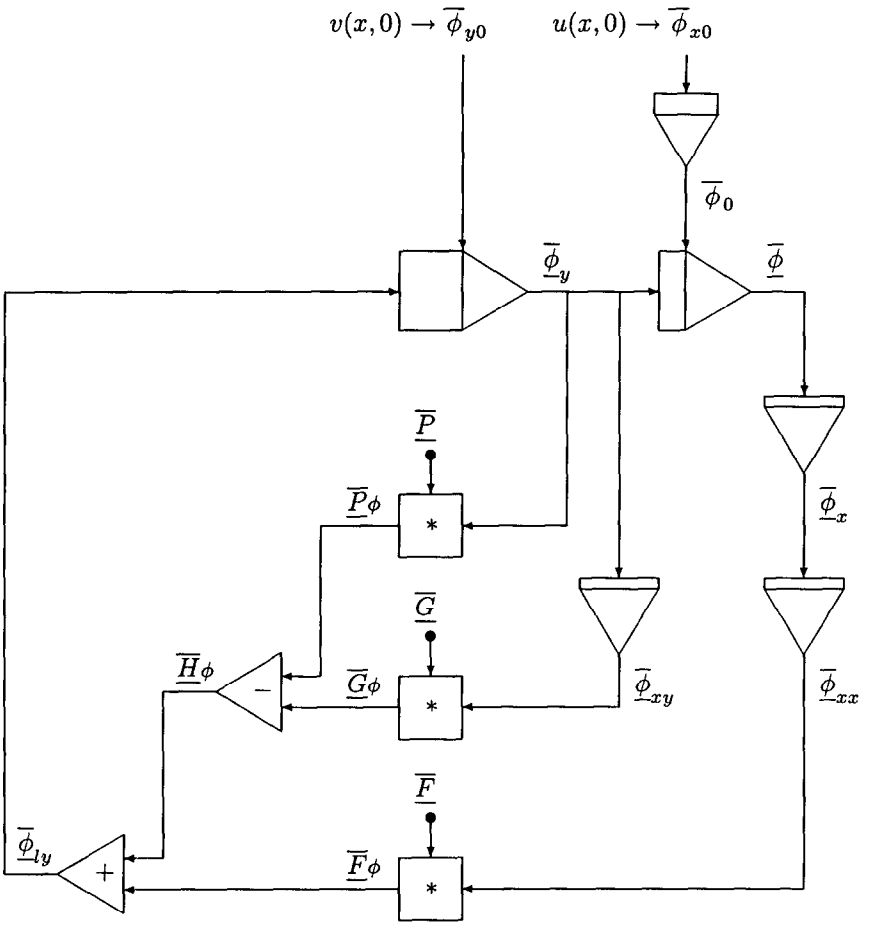


Figure 25. Main Analog Diagram of $\overline{\phi}_{yy} + \overline{\phi}_y \neq \overline{Y} = \overline{F} * \overline{\phi}_{xx} + \overline{G} * \overline{\phi}_{xy} - \overline{P} * \overline{\phi}_y$.

At the same time, the length in the x-direction becomes less in every loop due to the $\frac{\partial}{dx}$ differentiation, though only by one because multiplication with a leading zero in the leading column raises the pointer by one position again. An echelon form develops with the y-dimension about two × the x-dimension, the lower triangular part remaining empty (but still occupying computer memory). The bi-convolution numbers $\overline{\phi}$ and $\overline{\phi}_x$ are full after three loops (counted from 0) and $\overline{\phi}_y$ is filled after another loop.

The same pattern of occupied positions appears for bi-convolution numbers of any larger size, in fact, the occupancy of zeros and nonzeros in the columns shown, except the one containing the initial values, are exactly the same as the last columns of any larger size. Note however that the pattern depends on the first initial columns, for other initial conditions the pattern of occupied positions will be different.

The three terms of Sauer's solution, equation (165), are the same in this solution. Computation results are shown graphically in Figure 26(a). The bi-convolution number size was $n \times m = 17 \times 31$. The Mach-lines from 0.5 to 1.7 are shown, computed from the bi-convolution numbers $\overline{\phi}_x$ and $\overline{\phi}_y$, on a grid of y =const. lines from $x = -1$ to $+1$ and $y = 0$ to $+1$. The streamlines

are numerically integrated values of $dy = \frac{v}{1+u}dx$. The two characteristics emanating from $x, y = 0, 0$, obtained by numerical integration, are also shown.

All lines are drawn within the numerical region of convergence which is outlined graphically in Figure 26(b). Figure 26(b) shows the value of the error $f = L(\phi)$ from equation (167), determined from the bi-convolution numbers, and plotted superimposed on the (x, y) positions along the y-const. lines using a scale of 1. This carpet type plot indicates the numerical region of convergence graphically by the same method as in Section 19 for univariate Taylor series. There is hardly any region of gradually worsening approximation before the error lines fall off steeply.

Comparing with previous results by the method of [71], shown graphically in [65], the first obvious feature is that the region of convergence of these results is considerably less in the supersonic region, although the shape of the outer streamline through $x, y = 0, 0.5$ is more or less the same. The second striking feature is that compared to reported results [66,73], the streamline curvature within the region of convergence doesn't come anywhere near the value of $\frac{\rho_t}{y_t}$ (radius of curvature/throat) $= 0.5$.

To continue the solution, a new line $y = y_c > 0$, well within the region of convergence, is chosen and the solution started again. The transformed equation (169), with y replaced by $y + y_c$, is

$$\underline{\overline{\phi}}_{yy} = \underline{\overline{F}} * \underline{\overline{\phi}}_{xx} + \underline{\overline{G}} * \underline{\overline{\phi}}_{xy} - \underline{\overline{E}} * \underline{\overline{\phi}}_{y} \not\neq \underline{\overline{Y}}_c, \tag{174}$$

where now $\underline{\overline{Y}}$ is the Taylor transform of $y + y_c$,

$$\underline{\overline{Y}} = \begin{bmatrix} y_c & 1 & 0 & \cdots \\ 0 & & & \\ \vdots & & & \end{bmatrix}.$$

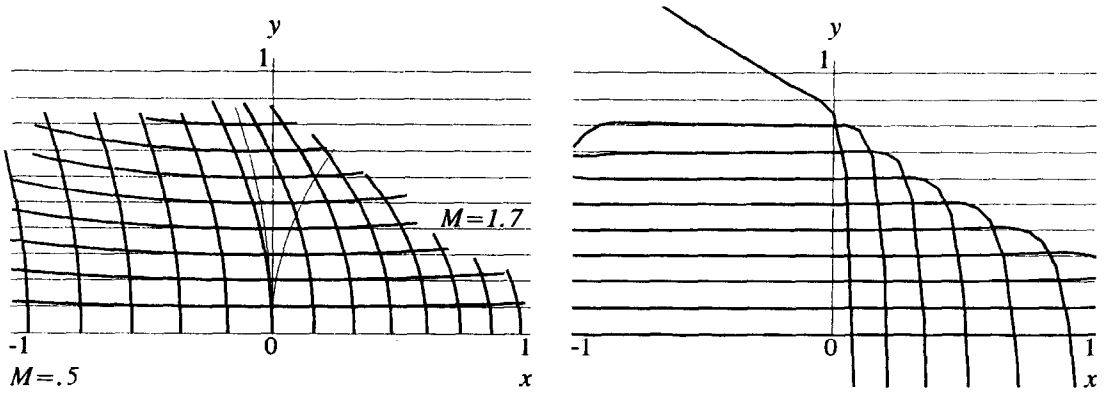
The initial convolution values are now, in computerstyle,

$$\text{new } \overline{\phi}_0 = \text{old } \underline{\overline{\phi}} \cdot \overline{y}_c,$$
$$\text{new } \overline{\phi}_{y0} = \text{old } \underline{\overline{\phi}}_y \cdot \overline{y}_c,$$

where the latter is not $\overline{0}$. In this equation, the solution loop starts with the integration of the term $\underline{\overline{\phi}}_{yy}$, the division by $\underline{\overline{Y}}_c$ is regular. The resulting bi-convolution numbers are upper triangular with no zero columns in between. Since the initial convolution values were given with length $n = 17$, the computation was made with size $n \times m = 17 \times 17$. The value of $y_c = 0.4$ was chosen as next initial line. The initial convolution value $\overline{\phi}_{y0}$ for the first integrator must be supplied from the previous computation, by simply computing $\underline{\overline{\phi}}_0 \cdot \overline{y}_c$. The next initial value $\overline{\phi}_0$, or $\overline{\phi}_{x0}$, must also be supplied by the previous calculation.

The results of the continuation are shown in Figure 27(a) and their convergence region indicated graphically as before in Figure 27(b). Contrary to any expectations, the region of convergence is not extended beyond the region of the first computation. It seems, therefore, that both are limited by the same singularities above the y-axis. Now the region of convergence is of course limited by the same distance below the new initial line as above, with the result that the region of convergence between $y = 0$ and y_c is much less than in the first computation.

To assess whether the new computation is a substantial improvement on the original method of Sauer (of 50 years ago, it was only published in the English language some five years later), Sauer's exact result of the approximate transonic equation, for the same initial values, is shown



(a) Machlines and stream lines. (b) Error function.

Figure 26. Transonic nozzle, base solution.

graphically in Figure 28(a) by way of Mach lines and streamlines as before, over the whole x, y region of the graph, and this can be compared to Figure 26(a). However, to asses the accuracy, the error calculation was made with the exact gas-dynamic equation (167) and is shown by means of the same carpet type plot in Figure 28(b). Whether such an approximation is acceptable depends on the application and the historical time. As it happens, the error is almost zero between the sonic line and the rearward characteristic, therefore giving a good example of a sonic line.

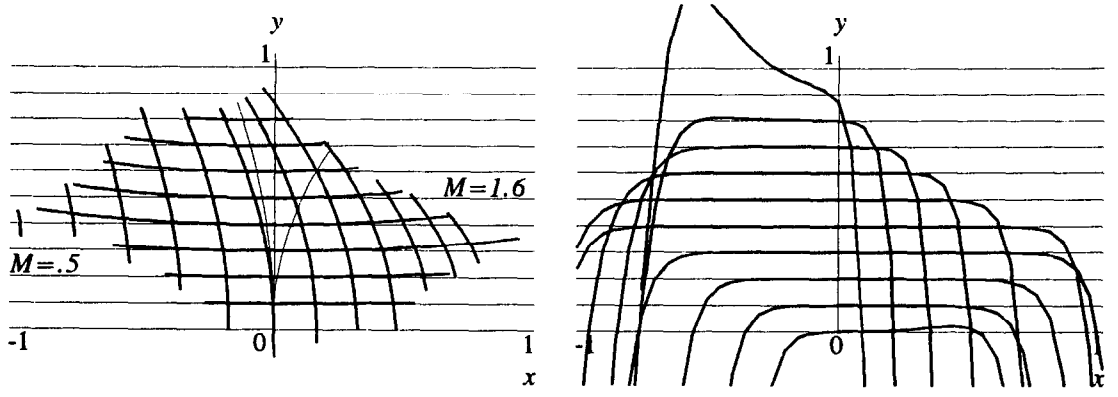
A purely subsonic case was computed with the initial condition $u(x, y) = -0.2 - 0.5x^2$ with size $n \times m = 17 \times 17$ bi-convolution numbers. The resulting bi-convolution numbers are upper triangular, but every second column and every second row is zero, so that only 1/4 of the triangle is occupied with numbers.

The results are shown in Figure 29(a) and their convergence region indicated graphically as before in Figure 29(b). They may be compared to similar subsonic solutions [65,71], although other trials of initial values may give better correspondence of the boundary.

The convergence boundaries are not very sharply defined, therefore, apparently better results can be obtained with larger bi-convolution numbers. The convergence boundaries of a computation with size $n \times m = 31 \times 31$ bi-convolution numbers is shown in Figure 30(b). The boundaries are much sharper defined as before by the steeper divergence. However, the useful region has hardly increased and the accuracy within the convergence region has not improved at all as the calculations showed, shown graphically in Figure 30(a). It appears, therefore, that within the seven digits on the IBM, PC accuracy is limited, further improvement cannot be obtained by larger bi-convolution numbers. But more important, the region of convergence is already well-defined by lesser size of numbers.

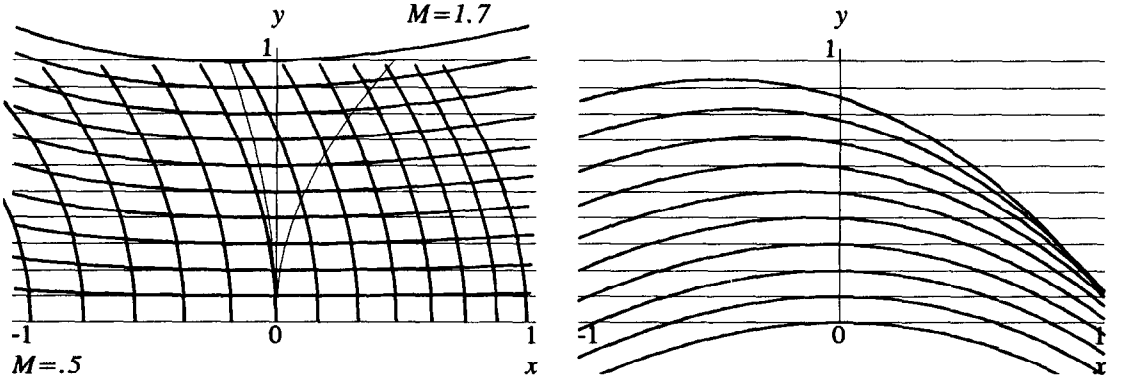
21.6. Partial Boundary Value Problem

The bi-convolution number suits the initial value problem naturally, but not the classic boundary value problem, for which a similar but symbolic method has been given by [19,20,23] Nevertheless, for completeness, we consider the isomorphism of the boundary value problem of the ordinary nonlinear differential equation of Section 17.4. For the nonlinear partial differential equation the values of the function or some partial derivatives are given on a complete initial and end line of either variable as a function, which again is expressed in terms of a convolution number, corresponding to initial and end number values of the ordinary differential equation. We call this appropriately a partial boundary value problem because the boundary values concern only one variable, and the rest of the boundary remains unspecified. This will now be demonstrated with the same equation (167) for transonic axisymmetric flow through a nozzle.



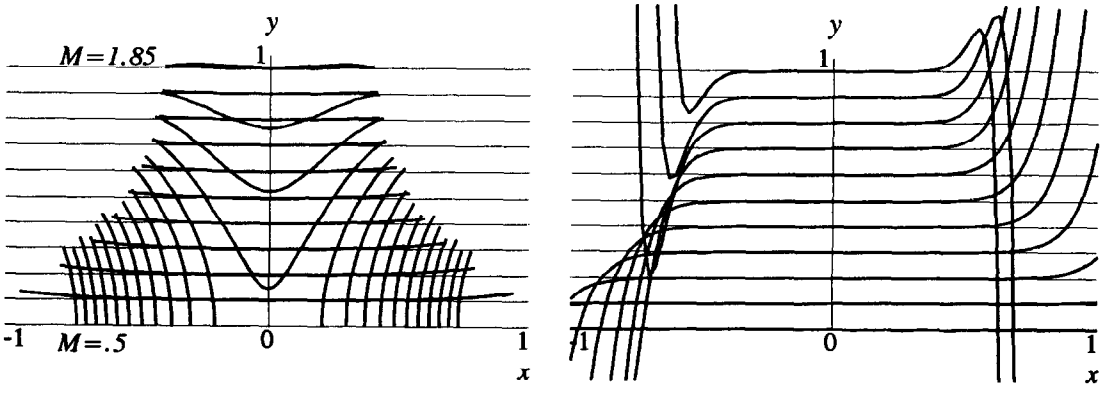
(a) Machlines and stream lines. (b) Error function.

Figure 27. Transonic nozzle, continued solution.



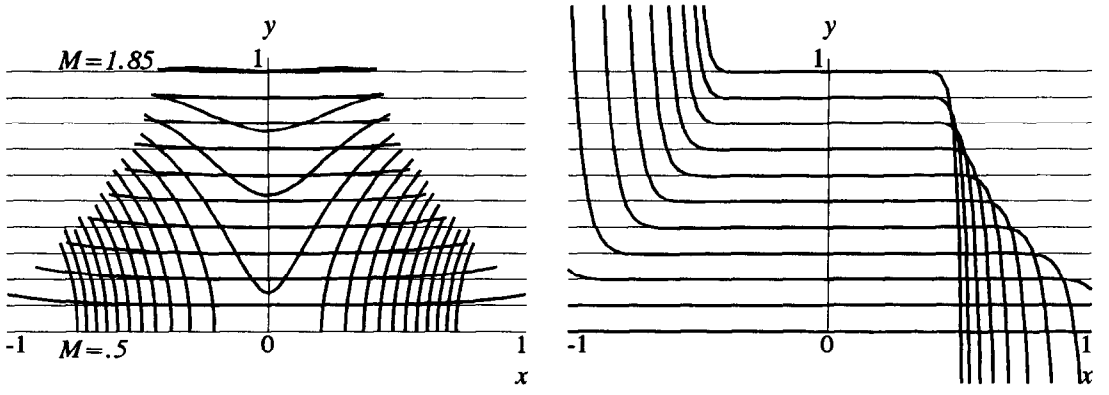
(a) Machlines and stream lines. (b) Error function.

Figure 28. Transonic nozzle, Sauer's solution.



(a) Machlines and stream lines. (b) Error function.

Figure 29. Transonic nozzle, 17 × 17 solution.



(a) Machlines and stream lines. (b) Error function.

Figure 30. Transonic nozzle, 31 × 31 solution.

The initial value is as before

$$v(x,0) = \phi_y(x,0) = 0$$
$$= \underline{x} \cdot \overline{v},$$

on the axis, $y = 0$. Let a parabolic throat boundary be specified by $y_b = y_t + \alpha x_b^2$ which in terms the function ϕ is specified as the gradient $g(x_b, y_b) = \dfrac{dy}{dx} = 2\alpha x_b = \dfrac{v(x_b, y_b)}{1 + u(x_b, y_b)}$. Consider the simpler boundary problem for the purpose of this illustation, *viz.* to specify the gradient on the line $y_b = \text{const.}$,

$$g(x) = 2\alpha x = \frac{v(x, y_b)}{1 + u(x, y_b)}$$
$$= \underline{x} \cdot \overline{g}.$$

To demonstrate the isomorphism further, we could write equation (168) as ordinary differential equation of the convolution variable $\overline{\phi}(y)$ as

$$\overline{b}(y) * \frac{d^2}{dy^2}\overline{\phi}(y) - \overline{a}(y) * \overline{\phi}_{xx}(y) - \overline{c}(y) * \frac{d}{dy}\overline{\phi}_x(y) + \overline{d}(y) * \overline{\phi}(y)/y = \overline{0}, \qquad (175)$$

with the two convolution numbers $\overline{v}$ at $y = 0$ and $\overline{g}$ at $y = y_b$ specified. This condition corresponds to the simpler case where only one initial number b and one final number c are specified for an ordinary differential equation of second order, and the scalar derivative $\dfrac{db}{dc}$ is required for the Newton-Raphson iteration. Note that in the derivation of equation (175) from equation (168), we have used identities like

$$\underline{x} \cdot \underline{\overline{D}} \cdot \overline{y} \times \underline{x} \cdot \underline{\overline{\phi}} \cdot \overline{y} = \underline{x} \cdot \left(\left(\underline{\overline{D}} \cdot \overline{y} \right) * \left(\underline{\overline{\phi}} \cdot \overline{y} \right) \right)$$
$$= \underline{x} \cdot \left(\overline{d}(y) * \overline{\phi}(y) \right),$$

using Theorem 1.

The bi-convolution equation (168) is now solved by the same Newton-Raphson iteration method, for which accordingly the Jacobian matrix $\frac{d\overline{g}}{d\overline{u}}$ must be found to iterate the unknown initial value $\overline{u}$. From

$$u(x_b, y_b) = \underline{x}b \cdot \underline{\overline{\phi}}_x \cdot \overline{y}b\,, \quad v(x_b, y_b) = \underline{x}b \cdot \underline{\overline{\phi}}_y \cdot \overline{y}b$$

follows

$$\begin{aligned}\overline{g} &= \frac{\underline{\overline{\phi}}_y}{1 + \underline{\overline{\phi}}_x} \\ &\equiv \underline{\overline{\psi}}.\end{aligned}$$

Therefore, the vector derivative is

$$\frac{d\overline{g}}{d\overline{u}} = \frac{d}{d\overline{u}}\left(\underline{\overline{\psi}} \cdot \overline{y}b\right) = \left[\frac{\partial\underline{\overline{\psi}}}{du_1}, \frac{\partial\underline{\overline{\psi}}}{du_2}, \cdots, \frac{\partial\underline{\overline{\psi}}}{du_k}\right] \cdot \overline{y}b, \tag{176}$$

with a dimension k to be found later, and where for each u_j the differential

$$d\underline{\overline{\psi}} = \frac{-\underline{\overline{\phi}}_y}{\left(1 + \underline{\overline{\phi}}_x\right)^{*2}} * d\underline{\overline{\phi}}_x + \frac{1}{1 + \underline{\overline{\phi}}_x} * d\underline{\overline{\phi}}_y.$$

Here, the bi-convolution differentials $d\underline{\overline{\phi}}_x$ and $d\underline{\overline{\phi}}_y$ are the Taylor transforms of the variations $\delta\phi_x$ and $\delta\phi_y$, respectively, while $\underline{\overline{\phi}}_x$ and $\underline{\overline{\phi}}_y$ are derivatives of the main solution equation (168). The equation for $\delta\phi$ is obtained from the varition of equation (167)

$$B\delta\phi_{yy} - A\delta\phi_{xx} - C\delta\phi_{xy} + D\frac{\delta\phi_y}{y} + \delta B\phi_{yy} - \delta A\phi_{xx} - \delta C\phi_{xy} + \delta D\frac{\phi_y}{y} = 0, \tag{177}$$

where the variations of the coefficients are, from equation (166),

$$\begin{aligned}\delta B &= -\left((\gamma-1)\delta\phi_x + (\gamma-1)\phi_x\delta\phi_x + (\gamma+1)\phi_y\delta\phi_y\right), \\ \delta A &= (\gamma+1)\delta\phi_x + (\gamma+1)\phi_x\delta\phi_x + (\gamma-1)\phi_y\delta\phi_y, \\ \delta C &= 2(1+\phi_x)\delta\phi_y + 2\delta\phi_x\phi_y, \\ \delta D &= -\left((\gamma-1)\delta\phi_x + (\gamma-1)\phi_x\delta\phi_x + (\gamma-1)\phi_y\delta\phi_y\right).\end{aligned}$$

Equation (177) now becomes an equation of the form similar to equation (167),

$$\delta\phi_{yy} + e_0\frac{\delta\phi_y}{y} = F\delta\phi_{xx} + G\delta\phi_{xy} - P\delta\phi_y + K\delta\phi_x + L\delta\phi_y, \tag{178}$$

of which the Taylor transform is the bi-convolution differential equation

$$d\underline{\overline{\phi}}_{yy} + \overline{e}_0 * d\underline{\overline{\phi}}_y \not\equiv \underline{\overline{Y}} = \underline{\overline{F}} * d\underline{\overline{\phi}}_{xx} + \underline{\overline{G}} * d\underline{\overline{\phi}}_{xy} - \underline{\overline{P}} * d\underline{\overline{\phi}}_y + \underline{\overline{K}} * d\underline{\overline{\phi}}_x + \underline{\overline{L}} * d\underline{\overline{\phi}}_y. \tag{179}$$

This equation is solved with pointer numbers by the same routine as equation (170) according to analog diagram Figure 25, with only the two terms with $\underline{\overline{K}}$ and $\underline{\overline{L}}$ added. The difference,

however, is that equation (179) is linear, with constant bi-convolution coefficients $\underline{F}$, $\underline{G}$, $\underline{P}$, $\underline{K}$, $\underline{L}$, which are computed from the main solution $\underline{\phi}$ during each iteration loop.

From physical reasoning it is clear that u_0, the leading element of $\underline{u}$, must also be specified as boundary value for a well posed problem, i.e., some velocity for the flow in the nozzle must be specified. The solutions $d\underline{\phi}$ are linearly dependent on the initial conditions $\underline{u}$, therefore, l independent solutions are found by the l initial conditions

$$\underline{u} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix}, \quad \cdots \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

For these initial conditions the solutions are already identical to the partial derivatives $\dfrac{d\underline{\phi}}{du_j}$, and $\dfrac{d\underline{\phi}_x}{du_j}$ and $\dfrac{d\underline{\phi}_y}{du_j}$ are then available from the same solution loop. After each calculation one column of the Jacobian matrix of equation (176) is computed

$$\frac{d\underline{g}}{du_j} = \left(\frac{1}{1+\underline{\phi}_x} * \frac{\partial \underline{\phi}_y}{du_j} - \frac{\underline{\phi}_y}{(1+\underline{\phi}_x)^{*2}} * \frac{\partial \underline{\phi}_x}{du_j} \right) \cdot \underline{y}_b. \tag{180}$$

The total iteration program consists of the following steps:

- $\underline{v} = \underline{0}$, specify y_b, and specify

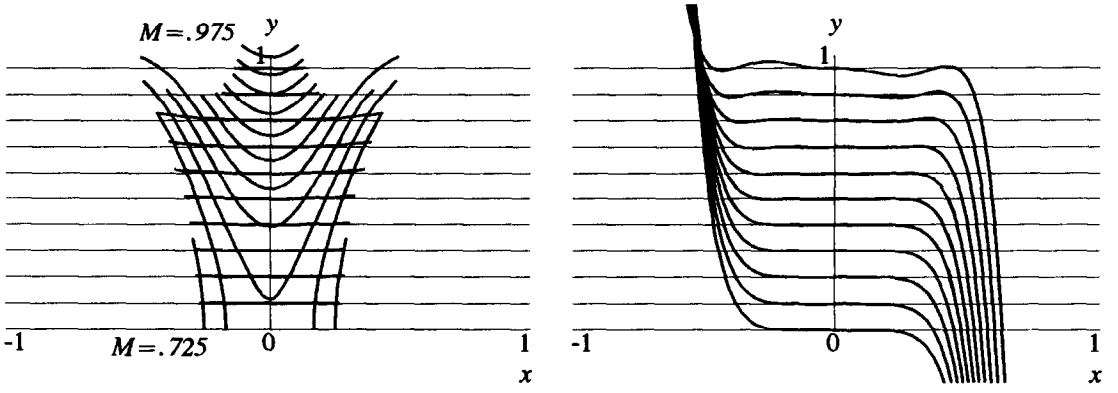
$$\underline{g} = \begin{bmatrix} 0 \\ 2\alpha \\ 0 \\ \vdots \end{bmatrix}.$$

- Assume some initial $\underline{u}$.
- Solve the main equation (170) for $\underline{\phi}$ and its partial derivatives.
- For each $\underline{u}_j$, solve equation (179) using the results of above for the bi-convolution constants.
- Compute one column of $\dfrac{d\underline{g}}{d\underline{u}}$ according to equation (180).
- Repeat l times until the Jacobian Matrix $\dfrac{d\underline{g}}{d\underline{u}}$ is complete.
- Compute the iterate

$$\underline{g}_{it} = \frac{\underline{\phi}_y}{1+\underline{\phi}_x} \cdot \underline{y}_b.$$

- The difference $d\underline{g} = \underline{g}_{it} - \underline{g}$.
- Compute the correction by solving for $d\underline{u}$, elements 1 to l, from the matrix equation

$$\frac{d\underline{g}}{d\underline{u}} \cdot d\underline{u} = d\underline{g}.$$



(a) Machlines and stream lines. (b) Error function.

Figure 31. Subsonic nozzle, iterated solution.

- Correct: new $\overline{u}$ = old $\overline{u} - d\overline{u}$, for Elements 1 to l.
- Repeat new main solution and correction until $d\overline{u}$ or $d\overline{g}$ is small enough.

The result of a calculation with size $n \times m = 7 \times 7$ bi-convolution numbers is shown in Table 3 below. The initial estimate of $\overline{u}$ is shown in Column 1, for the specified $\overline{g}$ in Column 4. Convergence within .000001 was reached in 13 iterations. The corresponding graphical representation of the flow is shown in Figure 31(a) and the estimated region of convergence in Figure 31(b). The calculation in Section 21.5 showed that the effective length of $\overline{u} = \overline{\phi}_{x0}$ is $n - 1$, therefore, the dimension k in the Jacobian matrix of equation (176) is $k = 6$.

As it turns out, this problem is not well-posed either. With a size $n \times m = 11 \times 11$ bi-convolution number convergence could not be reached. Whether a solution can be obtained or not is unfortunately not evident from the manner of solution. An indication is the Jacobian which becomes ill-conditioned tending to singular when the boundary value problem is ill-posed. The same problem for the transition flow from subsonic to supersonic flow, with the initial convolution value $\overline{u}$ from the previous example in Figure 29, has an extremely ill-conditioned Jacobian and does not converge at all. On the other hand, a boundary value of $\overline{v}$ can be specified for subsonic-supersonic flow but not for subsonic flow.

Table 3. Iterated initial and final boundary convolution values.

i	initial $\overline{u}$	final $\overline{u}$	initial $\overline{g}$	final $\overline{g}$
0	−0.2	−0.2	0	0
1	0	0	0.10126	0.2
2	−0.5	−0.79625	0	0
3	0	0	0.26183	0
4	0	1.95567	0	0
5	0	0	0.09181	0
6	0	−7.91187	0	0
7	0	0	0	0

Whether the bi-convolution number can be useful in partial or complete boundary value problems remains to be seen, but it is certainly not designed for that purpose. The recursive solution theory is based on the fact that a bi-convolution number is considered as a convolution number of a convolution number in either of two ways. Accordingly, the integration of a partial differential equation can only proceed in one of the two variables, i.e., the y-variable in the example

of transonic flow. In the other, variable differentiation is performed. The problem is that in the iteration solution $\overline{\underline{\phi}}$ and its derivatives are not true bi-convolution numbers. Firstly, by the manner of computation using the Jacobian matrix, the dependence of all elements is mixed and Theorem 4 does not apply. Secondly, the elements are not computed by a finite number of steps. The computed result may not be the matrix of coefficients of a double Taylor series at all but rather an approximation in the sense of Weierstrass, beyond the region of convergence. The graphical representation of accuracy in Figure 31(b) may, therefore, also not represent a convergence boundary, a true test would have to be made in the complex space.

Also the iteration is expensive, using $1 + n\times$ *iterations* as many computations as an initial value solutions.

22. CONVOLUTION VARIABLES WITH FUNCTIONAL ELEMENTS

Just as bi-convolution numbers were developed by putting convolution numbers as elements into a convolution number, we can put functions as elements into the convolution number. The result is of course not a number any more but a generalized convolution variable, which we call a functional convolution variable—not to be confused with the convolution variable as defined in Section 7. In the context of this section we use the term convolution variable as default.

The Taylor transformation equation (151) of Section 21.1 of the bivariate function $a(x, y)$ actually consists of two transformations which may be applied in any one of two sequences. If we apply only one of the transformations, e.g.,

$$\overline{\underline{X}}{}^{n}_{x} \cdot \overline{\underline{F}}{}^{x}_{y} = \overline{\underline{F}}{}^{n}_{y}, \tag{181}$$

then we obtain a partial transform, which is similar to the half transformation of a tensor in two different bases, which here is base n with countably infinite dimensions and base y with continuous infinte dimensions. We may appropriately call this a partial transform, obtained by a partial transformation, because it involves only one variable, similar to partial differentiation. Such a convolution variable was actually mentioned in Section 21.6.

The inverse of equation (181) is written

$$\overline{\underline{X}}{}^{x}_{n} \cdot \overline{\underline{F}}{}^{n}_{y} = \overline{\underline{F}}{}^{x}_{y}, \tag{182}$$

where $\left[\overline{\underline{X}}{}^{n}_{x}\right]^{-1} = \overline{\underline{X}}{}^{x}_{n}$. More simply we write the partial Taylor transformation,

$$\overline{x}^{-1} \cdot f(x, y) = \overline{f}(y) \tag{183}$$

and its inverse

$$\underline{x} \cdot \overline{f}(y) = f(x, y), \tag{184}$$

as default for the completely notated transformation equations (181) and (182), respectively. Here,

$$\overline{f}(y) = \{f_0(y),\ f_1(y),\ f_2(y) \ldots\} \tag{185}$$

is the functional convolution variable. It is not only a sequence of functions, it must be connected to a Taylor series by equation (184), i.e., it is a sequence of functional coefficients.

None of the numerical convolution number routines of the previous sections can be employed any more, but the convolution algebra formulas are the same and can be employed symbolically.

They are widely used to generate sequences of equations to solve nonlinear partial differential equations in two variables, e.g., as alternative method for the transonic equation of Section 21.5, boundary layer equations [57], but also to solve nonlinear ordinary differential equations by perturbation methods [57,76]. In the next subsection, we merely show how these expansions are written in terms of convolution variable notation, which we then use as a tool to show how all the convolution theorems apply.

Generally, there is an infinite sequence of further convolution variables that can be defined, e.g., a convolution number with bivariate functions as elements, or a bi-convolution number with functional elements etc. For any particular problems such numbers will have to be designed appropriately.

22.1. Functional Convolution Variable Notation

The notation of algebra with functional covolution variables will be illustrated by examples, the first of which is the equation of the streamfunction ψ of the laminar boundary layer of a quasi free stream velocity field $u(x)$, [57],

$$\psi_y\psi_{xy} - \psi_x\psi_{yy} = uu' + \nu\psi_{yyy}, \tag{186}$$

where $u = \psi_y(x,0)$. The solution is attempted in the form of a Taylor series in x with functional coefficients in y

$$\begin{aligned}\psi(x,y) &= \psi_0(y) + x\psi_1(y) + x^2\psi_2(y)\ldots \\ &= \underline{x}\cdot\overline{\psi}(y).\end{aligned}$$

The free stream variable $u(x,y) = \underline{x}\cdot\overline{u}$ is transformed to a convolution constant

$$\overline{u} = \{u_0,\, u_1,\, u_2,\, u_3\ldots\},$$

which, however, in the notation of Section 12, has x-derivatives $u'(x,y) = \underline{x}\cdot\overline{u}'$, where

$$\overline{u}' = \{u_1,\, 2u_2, 3u_3\ldots\}.$$

The function ψ and its x- and y- derivatives are written

$$\begin{aligned}&\psi(x,y) = \underline{x}\cdot\overline{\psi}(y), \qquad \psi_x(x,y) = \underline{x}\cdot\overline{\psi}'(y), \qquad \psi_y(x,y) = \underline{x}\cdot\overline{\psi}_y(y)\\ &\psi_{xy}(x,y) = \underline{x}\cdot\overline{\psi}'_y(y), \qquad \psi_{yy}(x,y) = \underline{x}\cdot\overline{\psi}_{yy}(y), \qquad \psi_{yyy}(x,y) = \underline{x}\cdot\overline{\psi}_{yyy}(y).\end{aligned}$$

The partial Taylor transform of $\psi(x,y)$ is

$$\overline{\psi}(y) = \{\psi_0(y),\, \psi_1(y),\, \psi_2(y)\,\psi_3(y)\ \ldots\},$$

and, therefore, the x-derivative is immediately obtained

$$\overline{\psi}'(y) = \{\psi_1(y),\, 2\psi_2(y)\ldots\}.$$

Substituting into equation (186), we obtain, using Theorem 1,

$$\overline{\psi}_y(y) * \overline{\psi}'_y(y) - \overline{\psi}'(y) * \overline{\psi}_{yy}(y) = \overline{u} * \overline{u}' + \nu\overline{\psi}_{yyy}. \tag{187}$$

In the original [57], f is written for the sequence of functions $\overline{\psi}$. Applying convolution algebra to equation (187), the familiar sequence of differential equations for the functions $\psi_i(y)$ are generated. Note that the constants $u_1, u_2, \ldots$ in [57] are not required, they merely represent

the values of the functions $\psi_1(0)$, $\psi_2(0)$,... but are used there for a further expansion. The partial Taylor transformation and convolution algebra have converted the partial differential equation (186) into a sequence of ordinary differential equations, similar to the application of the Laplace or Fourier transform to a partial differential equation, albeit on a different level.

As the next example, consider a nonlinear ordinary differential equation from vibration theory [76]

$$\frac{d^2u}{dx^2} + u^3 = 0. \tag{188}$$

The quantity $u(x)$ is of course not a function of two variables, nevertheless, perturbation theory expands

$$u(x) = u_1(x) + u_2(x) + u_3(x) + \cdots \tag{189}$$

in a sequence of functions of, hopefully, decreasing magnitude. When the expansion equation (189) is substituted into equation (188), only one equation in one variable results, which is then artificially separated into equations of different magnitude, to which there is no theoretical basis whatsoever. The sound practical technique, however, results in an iterative approximation by adding successively smaller corrections to the first solution. The technique has been made formal by introducing a "bookkeeping" parameter, or device [75], so that equation (189) appears as Taylor expansion

$$u(x) = u_0(x) + \epsilon u_1(x) + \epsilon^2 u_2(x) + \cdots, \tag{190}$$

corresponding to a function $u(\epsilon, x)$ of the two variables ϵ and x, without yet explaining the variable character of the bookkeeping parameter.

We can put the ordering technique on a sound theoretical basis by letting $u(\epsilon, x)$ be a continuous function of the continuous variables ϵ and x, in which the actual function $u(\epsilon = 1, x)$ is *embedded*, then the separation of the sum equation (190) into a sequence is mathematically sound.

It doesn't matter that there is still some arbitrariness where to put the quantity ϵ into the original equation. The situation may be compared to the equation that represents a single line in two-dimensional space, $\psi(x, y) = 0$. This function is not a unique description of the line, e.g., any regular function $f(\psi(x, y)) = 0$ represents the same line. The single line is now embedded in any of the functions $f(\psi(x, y))$ which may be chosen to suit the purposes of the problem. But many nonlinear equations have such a parameter ϵ already, e.g., the Mathieu equation $\ddot{u} + (a + \epsilon \cos 2t)u = 0$, which can then be considered as a continuous variable.

In our notation, the partial Taylor transform of equation (188) with respect to the continuous variable ϵ is

$$\overline{u}_{xx}(x) + \overline{u}^{*3}(x) = \overline{0}. \tag{191}$$

Generally, in perturbation theory u_0 is taken as zero, but this is just a convenience for further development, not a necessity to create a sequence of differential equations, which are obtained by completing the convolution algebra of equation (191).

The fact that perturbation theory often works outside the radius of convergence of ϵ does not affect the application of convolution theory.

The inclusion of perturbation theory of functions of one variable in the analysis of functions of two variables as done here is perhaps masked by the fact that perturbation theory transforms differential equations not to the general form of partial differential equations because no derivatives in ϵ occur. Therefore also initial values along the ϵ axis are not required.

22.2. Functional Convolution Variable Theory

Functional convolution variables are actually outside the scope of this treatise, whose purpose is to develop number routines for numerical computations. Therefore we will only show how the convolution theory applies to these quantities.

Having identified the sequence of functions in equations (185) and (189) as convolution variable, immediately all the Theorems 1–6 apply to functions $\overline{f}(\overline{u})$ and to differential operators $\overline{L}(\overline{u})$, most of which are well-known from observation but not formally proved.

Theorem 1 defines the transformations of algebraic operations which were already discussed above.

Theorem 2 states the compatibilty equations from which the practically important conclusions follow which explain the recursive method of solution. Because the u_j in the convolution variable $\overline{u}$ are now functions, the derivative elements of the Jacobian matrix equation (68) must be seen as variationals $\dfrac{\delta f_i}{\delta u_j}$.

The first conclusion is Theorem 3, stating that the first function equation f_0 contains the element $u_0(y)$ only and is the same as the original equation in the function domain, which may be nonlinear, and may have several solutions. Theorem 4 states that the function element f_j contains the highest convolution element $u_j(y)$ only linear, and all other elements in that function are lower elements $u_{k<j}$, from which follows both the recursive solution of nonlinear equations and the fact that the solutions are obtained by a linear equation, therefore are unique. Theorem 5 states that there are not more distinct convolution solutions possible than the number of roots of the first equation of the first element.

Theorem 6 defines the transformation of derivatives, but in functional convolution derivatives, we must distinguish between differential and integral operations on x and those on y, the first and second variables, respectively, in as defined by $\underline{x} \cdot \overline{u}(y)$. Such operations on x shift the pointer as described in Section 4, while operations on y don't affect the pointer but create a new number $\overline{u}_y$ or $\overline{u}_Y$, whose elements are not related algebraically to the elements of $\overline{u}$. The differential operator in the convolution domain refers then only to differentiation and integration in x, so that the transform of a partial differential equation is written formally such as $\overline{L}(\overline{u}_{yy}, \overline{u}_y, \overline{u}) = 0$.

The implication in the theorems is that the first equation $L_0 = 0$ may contain the first element and all its derivatives, which may therefore be an ordinary nonlinear differential equation in y, and in all subsequent equations $L_i = 0$ the highest elements of all variables appear linearly according to Section 11.4, which are now u_i and any of its derivatives. Therefore, u_i appears in a linear ordinary differential equation in y, with any lower elements $u_{k<i}$ and their derivatives as coefficients and functions.

The pointer is shifted as in the ordinary convolution number by integration and differentiation in x, and by leading zeros in algebraic routines. The effect of the pointer on the positions of elements in the theorems is the same as for ordinary convolution numbers as described in Section 14. Particularly this means that in perturbation equations, even though no differentiation and integration in ϵ occurs, the conclusions of Theorem 4 apply, modified by compatible operations as explained in Section 14.1. The result is that elements $u_j(y)\ldots$ may be contained in an expression and connected to ϵ^k in the function domain where $k \neq j$. The pointer method may then be employed to trace in which positions the highest determined element $u_j(y)$ is, and in which sequence a recursive solution is possible.

23. CONCLUSION

The definition of the convolution number has created a theory that puts the previously well-known recursive solution methods for Taylor series on a sound theoretical basis. At the same time, a numerical algebra and analysis is created that is very easy to use similar to finite difference

or matrix methods. Recursive solution is made possible with the introduction of the pointer in the arithmetic and differential operation routines.

A convolution notation has been developed that makes convolution equations look as original equations in the number domain, yet with symbols that distinguish clearly between the two.

Complicated nonlinear differential equations are no more difficult to solve than simple linear problems. New functions are easily computed from their definition rather than reduced to a combination of other functions, such that the expression "a closed form solution" loses its exclusive meaning. The advantage of obtaining machine exact Taylor coefficients may be very useful theoretically.

Since numerical integration of ordinary differential equations is well established and cheap on the digital computer, the advantage of the convolution number lies not so much in obtaining numerical solutions, but rather to obtain a semi-analytic solution. Particular advantage may be applications to functions of two variables, which are complex functions for the ordinary convolution number and bivariate functions for the bi-convolution number. The usefulness is for initial value problems, the advantage of simple computations is quickly destroyed in boundary value problems, full boundary value problems in two dimensions haven't even been attempted.

The biggest drawback is the often very small radius of convergence in the complex space of functions which are perfectly regular in a real region.

The method does need much computer storage relative to the magnitude of a problem. Just like seven to eight digits have evolved as satisfactory for general computations on main computers, it may be that 14 to 16 digits or more will be the most satisfactory for convolution numbers, like some comparisons have shown. On present main computers this corresponds to double precision.

Within these limitations, this new computational tool may be valuable in many specific cases.

APPENDIX 1

The position of the Taylor transform in the family of transforms can be appreciated as follows. We may classify transforms in three levels, and in each level in two modes. The first, highest level is that of continuous functions which are transformed into continuous functions by the classic transforms, one mode is the Fourier transform which maps an infinite strip on an infinite strip, the other is the Laplace transform which maps a half-infinite plane on a half-infinite plane.

The second, intermediate level, is that of the transform of a continuous function on a countably infinite number of points. The one mode is the mapping of the points from negative infinity to positive infinity on the real axis on a circular strip by the Laurent series, and its inverse by the Cauchy integral formula. When the circular disk is mapped on a rectangular strip, the transformation becomes a discrete-continuous Fourier transform, i.e., the Fourier series and its inverse. The other mode is the transformation of the infinite number of points from only the half infinite real axis on the inside of a circular disk by the Taylor series, and its inverse which we now call the Taylor transform, also by the Cauchy integral formula. If the mapping to the infinite region outside the cicular disk is made, the transform is called the z-transform. The transforms on this level constitute the numerical-analytical interface. They allow us to perform operations on analytical functions by means of numerical computations with their coefficients. The Taylor and the z-transform are essentially inverses of each other, but the z-transform is interpreted and used differently, not as a numerical instrument to handle analytic functions, but rather the opposite.

The third, lowest level is the transformation of a finite number of points on a finite number of points by the discrete-discrete Fourier transform, the two modes are distinguished by whether the points are used for interpolation of a Taylor series or a Laurent series.

APPENDIX 2

Table A2. Beam System Matrix $\overline{\overline{Sn}}$ in slices $r = 0$, $r = 4$, $r = 8$ corresponding to $\omega^0\,\omega^2$, ω^4, and the slices $r = 15$, $r = 16$ containing l and m, respectively.

	Slice r = 0, (stiffness)			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	12	6	−12	6
2	6	4	−6	2
3	−12	−6	12	−6
4	6	2	−6	4

	Slice $r = 4$, ω^2 (mass)			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	−.3714286	−5.238095E-02	−.1285714	3.095239E-02
2	−5.238095E-02	−9.523802E-03	−3.095239E-02	7.142872E-03
3	−.1285715	−3.095239E-02	−.3714286	5.238095E-02
4	3.095239E-02	7.142860E-03	5.238096E-02	−9.523805E-03

	Slice $r = 8$, ω^4			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	−3.648736E-04	−7.661665E-05	−3.295727E-04	7.219252E-05
2	−7.661653E-05	−1.626241E-05	−7.219298E-05	1.570403E-05
3	−3.295716E-04	−7.219308E-05	−3.648737E-04	7.661645E-05
4	7.219311E-05	1.570411E-05	7.661665E-05	−1.626238E-05

	Slice $r = 15$, pointer l			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

	Slice $r = 16$, pointer m			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	12	14	14	14
2	14	14	14	14
3	8	8	8	8
4	12	12	12	12

APPENDIX 3

Table A3. 4 DOF Transfer Function Convolution Matrix $\overline{\overline{H}}_P$ in slices $r = 0$ to 6 Divsionless Gauss Elimination Method.

	Slice $r = 0$, Real			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	4000000	3000000	2000000	1000000
2	3000000	6000000	4000000	2000000
3	2000000	4000000	6000000	3000000
4	1000000	2000000	3000000	4000000

	Slice $r = 1$, Imag.			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	12700	9200	5999.999	3000
2	9200	18400	12000	6000
3	6000	12000	18400	9200
4	3000	5999.999	9199.999	12700

	Slice $r = 2$, Real			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	−130013.4	−60009.4	−20006	−3.001875
2	−60009.4	−180018.8	−80011.99	−20006
3	−20006	−80011.99	−180018.8	−60009.41
4	−3.001003	−20006	−60009.4	−130013.4

	Slice $r = 3$, Imag			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	−272.0046	−122.0032	−40.00198	−9.985352E-04
2	−122.0032	−368.0063	−160.004	−40.00199
3	−40.00196	−160.004	−368.0063	−122.0031
4	−1.013333E-03	−40.00204	−122.0032	−272.0046

	Slice $r = 4$, Real			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	1000.142	200.0621	2.009766E-02	0
2	200.0621	1600.188	400.0801	.019975
3	.020025	400.0801	1600.188	200.0623
4	0	2.005333E-02	200.062	1000.143

	Slice $r = 5$, Imag			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	1.04	.1999995	0	0
2	.2000004	1.639998	.4000004	0
3	0	.4	1.639998	.2000012
4	0	0	.2	1.04

	Slice $r = 6$, Real			
i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
1	−2.000003	0	0	0
2	0	−4	0	0
3	0	0	−4.000003	0
4	0	0	0	−1.999968

APPENDIX 4
DETAILS OF EXAMPLE 2

The convolution number expression equation (65) is

$$\overline{u} = \left(-\left(1 - \hat{c}\,\overline{\delta}_1\right) \pm \sqrt{\left(1 - \hat{c}\,\overline{\delta}_1\right)^2 + \overline{b} * \overline{\delta}_1} \right) / \overline{\delta}_1.$$

Presently, with each convolution number operation available as a separate subroutine, the equation (65) must be solved in steps of arithmetic operations, where each step is programmed by a call to the corresponding subroutine of convolution number arithmetic using intermediate variables $\overline{v}_i$.

Set up constants: $\overline{b}$, $\overline{v}_1 = 1 - \hat{c}\,\overline{\delta}_1$

$$\overline{v}_2 = \overline{v}_1^{*2}$$

$$\overline{v}_3 = \overline{b} * \overline{\delta}_1$$

$$\overline{v}_4 = \overline{v}_2 + \overline{v}_3$$

$$\overline{v}_5 = \sqrt{\overline{v}_4}$$
$$\overline{v}_6 = \overline{v}_1 + \overline{v}_5$$
$$\overline{u} = \overline{v}_6/\overline{\delta}_1.$$

The independent and dependent constants for the problem are

$$\begin{aligned} \psi_0 &= -.05, & c_x &= -.3587621, \\ \phi_s &= 0, & c_y &= .1278796, \\ \phi_1 &= -.1, & a &= .1857351, \\ \phi_2 &= .5, & \phi_c &= -.06853555. \end{aligned}$$

The stagnation points in the ζ-plane are

$$\begin{aligned} \zeta_s &= -.3420068 + \hat{\imath}\ .191799, \\ \zeta_{1s} &= -.1229288 - \hat{\imath}\ .1392337, \\ \zeta_{2s} &= .1829083 + \hat{\imath}\ .03228117. \end{aligned}$$

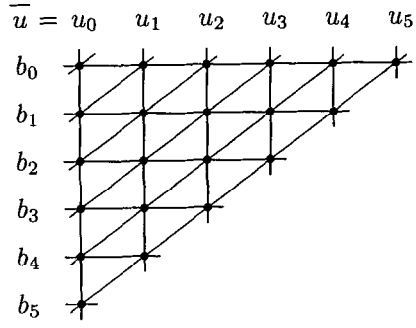
The convolution number $\overline{u}$ with length $n = 12$ computed on a IBM PC is

i	real part	imag. part
0	0	0
1	2.191074E-02	2.087835E-02
2	1.845746E-03	−2.768948E-04
3	−5.382224E-05	−1.220876E-04
4	−1.130089E-05	4.448460E-06
5	4.507057E-07	1.268736E-06
6	1.496934E-07	−4.862362E-08
7	5.241459E-09	−1.853280E-08
8	2.379207E-09	5.613774E-10
9	6.225862E-11	3.146268E-10
10	4.081696E-11	−6.793737E-12
11	−1.749080E-13	−5.656138E-12
12	0	0

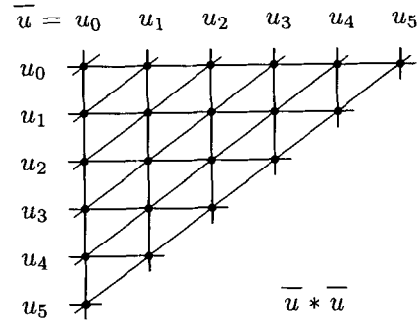
The zero in the last position is due to the division by $\overline{\delta}_1$. Convergence is not very good on the circle where the highest term of the series $|\zeta|^{11} = a^{11} = 9.074736E-09$. Accuracy is determined by using the exact mapping function of equation (58). The best accuracy was achieved with $n = 14$ with a lsq. error of .00186 referred to the radius a. For larger n the coefficients diverge quickly. Transforming the calculations to a unit circle doesn't change convergence and accuracy at all. It is the advantage of convolution number arithmetic that we do not have to transform to a particular radius to obtain best accuracy, since generally the radius of convergence is not known. The same calculations were done on a HP 9836 desk computer which has 13 digit arithmetic. The best accuracy was obtained with $n = 34$ with a lsq. error of .00001577 referred to a. In this example, we don't have any theoretical analysis to establish whether there is any type of singularity on the circle boundary. From the poor convergence, we can surmise that there is one at least close to the boundary and therefore we cannot achieve machine accuracy on the boundary, and apparently for the same reason we cannot determine the Taylor coefficents, i.e., the convolution number more accurately. This example illustrates that using Taylor series too near to the boundary of convergence is quite inefficient.

APPENDIX 5
DETAILS OF ANALYTICAL CONSTRUCTION OF THE CONVOLUTION FUNCTION, USING THE MULTIPLICATION ARRAY OF FIGURE 1

$$\overline{f}(\overline{u}) = \overline{a} + \overline{b} * \overline{u} + \overline{c} * \overline{u}^{*2} + \overline{d} * \overline{u}^{*3}$$
$$= \{f_0, f_1, f_2, f_3, f_4, f_5, \ldots\}$$

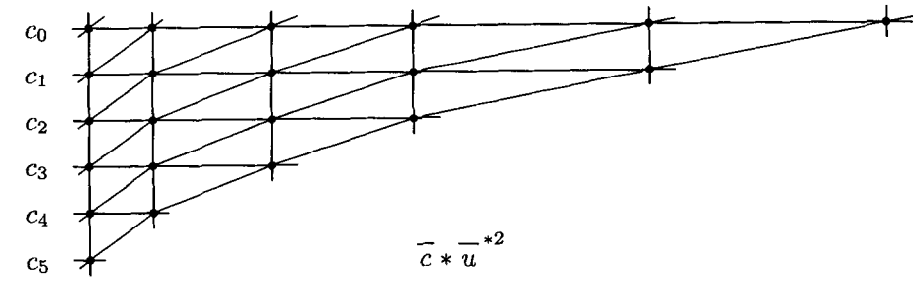


$$\overline{b} * \overline{u} = \begin{bmatrix} b_0u_0 \\ b_1u_0 + b_0u_1 \\ b_2u_0 + b_1u_1 + b_0u_2 \\ b_3u_0 + b_2u_1 + b_1u_2 + b_0u_3 \\ b_4u_0 + b_3u_1 + b_2u_2 + b_1u_3 + b_0u_4 \\ b_5u_0 + b_4u_1 + b_3u_2 + b_2u_3 + b_1u_4 + b_0u_5 \end{bmatrix}$$



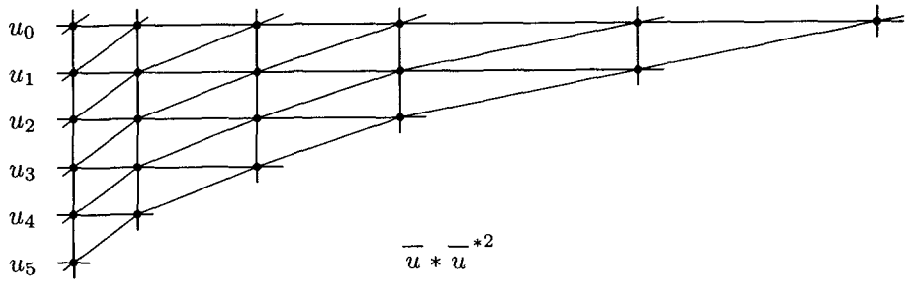
$$\overline{u}^{*2} = \begin{bmatrix} u_0^2 \\ 2u_0u_1 \\ 2u_0u_2 + u_1^2 \\ 2u_0u_3 + 2u_1u_2 \\ 2u_0u_4 + 2u_1u_3 + u_2^2 \\ 2u_0u_5 + 2u_1u_4 + 2u_2u_3 \end{bmatrix}$$

$$\overline{u}^{*2} = u_0^2 \quad 2u_0u_1 \quad 2u_0u_2 + u_1^2 \quad 2u_0u_3 + 2u_2u_1 \quad 2u_0u_4 + 2u_1u_3 + u_2^2 \quad 2u_0u_5 + 2u_4u_1 + 2u_2u_3$$



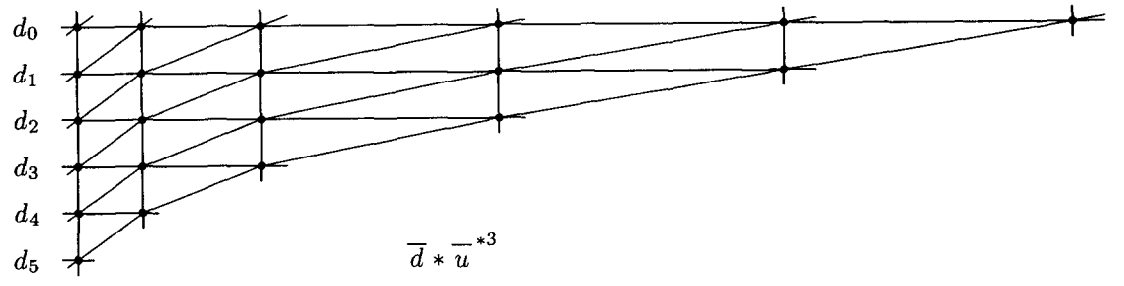
$$\overline{c} * \overline{u}^{*2} = \begin{bmatrix} c_0u_0^2 \\ c_1u_0^2 + 2c_0u_0u_1 \\ c_2u_0^2 + 2c_1u_0u_1 + 2c_0u_0u_2 + c_0u_1^2 \\ c_3u_0^2 + 2c_2u_0u_1 + 2c_1u_0u_2 + c_1u_1^2 + 2c_0u_0u_3 + 2c_0u_1u_2 \\ c_4u_0^2 + 2c_3u_0u_1 + 2c_2u_0u_2 + c_2u_1^2 + 2c_1u_0u_3 + 2c_1u_1u_2 + 2c_0u_0u_4 + 2c_0u_1u_3 + c_0u_2^2 \\ c_5u_0^2 + 2c_4u_0u_1 + 2c_3u_0u_2 + c_3u_1^2 + 2c_2u_0u_3 + 2c_2u_1u_2 + 2c_1u_0u_4 + 2c_1u_1u_3 + c_1u_2^2 \\ \qquad\qquad + 2c_0u_0u_5 + 2c_0u_1u_4 + 2c_0u_2u_3 \end{bmatrix}$$

$$\overline{u}^{*2} = u_0^2 \;\; 2u_0u_1 \;\; 2u_0u_2 + u_1^2 \;\; 2u_0u_3 + 2u_2u_1 \;\; 2u_0u_4 + 2u_1u_3 + u_2^2 \;\; 2u_0u_5 + 2u_4u_1 + 2u_2u_3$$



$$\overline{u}^{*3} = \begin{bmatrix} u_0^3 \\ 3u_0^2u_1 \\ 3u_0^2u_2 + 3u_0u_1^2 \\ 3u_0^2u_3 + 6u_0u_1u_2 + u_1^3 \\ 3u_0^2u_4 + 6u_0u_1u_3 + 3u_0u_2^2 + 3u_1^2u_2 \\ 3u_0^2u_5 + 6u_0u_1u_4 + 6u_0u_2u_3 + 3u_1^2u_3 + 3u_1u_2^2 \end{bmatrix}$$

$$\overline{u}^{*3} = u_0^3 \;\; 3u_0^2u_1 \;\; 3u_0^2u_2 + 3u_0u_1^2 \;\; 3u_0^2u_3 + 6u_0u_1u_2 + u_1^3 \;\; 3u_0^2u_4 + 6u_0u_1u_3 + 3u_0u_2^2 + 3u_1^2u_2 \;\; \ldots$$



$$\overline{d} * \overline{u}^{*3} = \begin{bmatrix} d_0u_0^3 \\ d_1u_0^3 + 3d_0u_0^2u_1 \\ d_2u_0^3 + 3d_1u_0^2u_1 + 3d_0u_0^2u_2 + 3d_0u_0u_1^2 \\ d_3u_0^3 + 3d_2u_0^2u_1 + 3d_1u_0^2u_2 + 3d_1u_0u_1^2 + 3d_0u_0^2u_3 + 6d_0u_0u_1u_2 + d_0u_1^3 \\ d_4u_0^3 + 3d_3u_0^2u_1 + 3d_2u_0^2u_2 + 3d_2u_0u_1^2 + 3d_1u_0^2u_3 + 6d_1u_0u_1u_2 + d_1u_1^3 \\ \quad + 3d_0u_0^2u_4 + 6d_0u_0u_1u_3 + 3d_0u_0u_2^2 + 3d_0u_1^2u_2 \\ d_5u_0^3 + 3d_4u_0^2u_1 + 3d_3u_0^2u_2 + 3d_3u_0u_1^2 + 3d_2u_0^2u_3 + 6d_2u_0u_1u_2 + d_2u_1^3 \\ \quad + 3d_1u_0^2u_4 + 6d_1u_0u_1u_3 + 3d_1u_0u_2^2 + 3d_1u_1^2u_2 \\ \qquad + 3d_0u_0^2u_5 + 6d_0u_0u_1u_4 + 6d_0u_0u_2u_3 + 3d_0u_1^2u_3 + 3d_0u_1u_2^2 \end{bmatrix}.$$

The addition is performed by collecting the terms of all multiplications, which are written here in a particular structural sequence

$$\begin{aligned}
f_0 &= a_0 + b_0u_0 + c_0u_0^2 + d_0u_0^3, \\
f_1 &= a_1 + b_1u_0 + c_1u_0^2 + d_1u_0^3 + b_0u_1 + 2c_0u_0u_1 + 3d_0u_0^2u_1, \\
f_2 &= a_2 + b_2u_0 + c_2u_0^2 + d_2u_0^3 + b_1u_1 + 2c_1u_0u_1 + 3d_1u_0^2u_1 + c_0u_1^2 + 3d_0u_0u_1^2 \\
&\quad + b_0u_2 + 2c_0u_0u_2 + 3d_0u_0^2u_2, \\
f_3 &= a_3 + b_3u_0 + c_3u_0^2 + d_3u_0^3 + b_2u_1 + 2c_2u_0u_1 + 3d_2u_0^2u_1 + c_1u_1^2 + 3d_1u_0u_1^2 + d_0u_1^3 \\
&\quad + b_1u_2 + 2c_1u_0u_2 + 3d_1u_0^2u_2 + 2c_0u_1u_2 + 6d_0u_0u_1u_2 \\
&\quad + b_0u_3 + 2c_0u_0u_3 + 3d_0u_0^2u_3, \\
f_4 &= a_4 + b_4u_0 + c_4u_0^2 + d_4u_0^3 + b_3u_1 + 2c_3u_0u_1 + 3d_3u_0^2u_1 + c_2u_1^2 + 3d_2u_0u_1^2 + d_1u_1^3 \\
&\quad + b_2u_2 + 2c_2u_0u_2 + 3d_2u_0^2u_2 + 2c_1u_1u_2 + 6d_1u_0u_1u_2 + 3d_0u_1^2u_2 + c_0u_2^2 + 3d_0u_0u_2^2 \\
&\quad + b_1u_3 + 2c_1u_0u_3 + 3d_1u_0^2u_3 + 2c_0u_1u_3 + 6d_0u_0u_1u_3 \\
&\quad + b_0u_4 + 2c_0u_0u_4 + 3d_0u_0^2u_4, \\
f_5 &= a_5 + b_5u_0 + c_5u_0^2 + d_5u_0^3 + b_4u_1 + 2c_4u_0u_1 + 3d_4u_0^2u_1 + c_3u_1^2 + 3d_3u_0u_1^2 + d_2u_1^3 \\
&\quad + b_3u_2 + 2c_3u_0u_2 + 3d_3u_0^2u_2 + 2c_2u_1u_2 + 6d_2u_0u_1u_2 + 3d_1u_1^2u_2 + c_1u_2^2 + 3d_1u_0u_2^2 + 3d_0u_1u_2^2 \\
&\quad + b_2u_3 + 2c_2u_0u_3 + 3d_2u_0^2u_3 + 2c_1u_1u_3 + 6d_1u_0u_1u_3 + 6d_0u_0u_2u_3 + 3d_0u_1^2u_3 \\
&\quad + b_1u_4 + 2c_1u_0u_4 + 3d_1u_0^2u_4 + 6d_0u_0u_1u_4 + 3d_0u_0^2u_5.
\end{aligned}$$

To prevent any misinterpretation, we emphasize that this algebra is not performed in numerical convolution number algebra where only sequences of numbers are used. The algebra here is only done for the theoretical basis of numerical programming.

By the same method we expand the convolution derivative

$$\begin{aligned}
\overline{g} &= d\overline{f}/d\overline{u} = \overline{b} + 2\overline{c} * \overline{u} + 3\overline{d} * \overline{u}^{*2} \\
&= \{g_0, g_1, g_2, g_3, g_4, g_5, \ldots\},
\end{aligned}$$

$$\begin{aligned}
g_0 &= b_0 + 2c_0u_0 + 3d_0u_0^2, \\
g_1 &= b_1 + 2c_1u_0 + 3d_1u_0^2 + 2c_0u_1 + 6d_0u_0u_1, \\
g_2 &= b_2 + 2c_2u_0 + 3d_2u_0^2 + 2c_1u_1 + 6d_1u_0u_1 + 3d_0u_1^2 + 2c_0u_2 + 6d_0u_0u_2, \\
g_3 &= b_3 + 2c_3u_0 + 3d_3u_0^2 + 2c_2u_1 + 6d_2u_0u_1 + 3d_1u_1^2 + 2c_1u_2 + 6d_1u_0u_2 + 6d_0u_1u_2 \\
&\quad + 2c_0u_3 + 6d_0u_0u_3,
\end{aligned}$$

$$g_4 = b_4 + 2c_4u_0 + 2d_4u_0^2 + 2c_3u_1 + 6d_3u_0u_1 + 3d_2u_1^2 + 2c_2u_2 + 6d_2u_0u_2 + 6d_1u_1u_2 + 3d_0u_2^2,$$
$$+ 2c_1u_3 + 6d_1u_0u_3 + 6d_0u_1u_3 + 2c_0u_4 + 6d_0u_0u_4,$$
$$g_5 = b_5 + 2c_5u_0 + 3d_5u_0^2 + 2c_4u_1 + 6d_4u_0u_1 + 3d_3u_1^2 + 2c_3u_2 + 6d_3u_0u_2 + 6d_2u_1u_2 + 3d_1u_2^2$$
$$+ 2c_2u_3 + 6d_2u_0u_3 + 6d_1u_1u_3 + 2c_1u_4 + 6d_1u_0u_4.$$

The second convolution derivative is

$$\overline{h} = d^{*2}\overline{f}/d\overline{u}^{*2}) = d\overline{g}/d\overline{u} = 2\overline{c} + 6\overline{d} * \overline{u}$$
$$= \{h_0, h_1, h_2, h_3, h_4, h_5, \ldots\},$$

$$h_0 = 2c_0 + 6d_0u_0,$$
$$h_1 = 2c_1 + 6d_1u_0 + 6d_0u_1,$$
$$h_2 = 2c_2 + 6d_2u_0 + 6d_1u_1 + 6d_0u_2,$$
$$h_3 = 2c_3 + 6d_3u_0 + 6d_2u_1 + 6d_1u_2 + 6d_0u_3,$$
$$h_4 = 2c_4 + 6d_4u_0 + 6d_3u_1 + 6d_2u_2 + 6d_1u_3 + 6d_0u_4,$$
$$h_5 = 2c_5 + 6d_5u_0 + 6d_4u_1 + 6d_3u_2 + 6d_2u_3 + 6d_1u_4 + 6d_0u_5.$$

APPENDIX 6
EXAMPLE OF MULTIROOT SOLUTION

For the purpose of demonstration of the solution of a nonlinear convolution equation with multiple root u_0 by the method of advance pointer convolution number, we construct a fourth order polynomial of $u(x)$ with known functions as roots

$$f = (u - (2 + \tan x)(u - \cos x)\left(u - \sqrt{1 - x}\right)\left(u - \frac{1}{1 + x}\right). \tag{A6.1}$$

The Taylor transform is the convolution polynomial

$$\overline{f} = \left(\overline{u} - \overline{a}\right)\left(\overline{u} - \overline{b}\right)\left(\overline{u} - \overline{c}\right)\left(\overline{u} - \overline{d}\right)$$
$$= \overline{s} + \overline{r} * \overline{u} + \overline{q} * \overline{u}^{*2} + \overline{p} * \overline{u}^{*3} + \overline{u}^{*4}, \tag{A6.2}$$

where

$$\overline{s} = \overline{a} * \overline{b} * \overline{c} * \overline{d}$$
$$\overline{r} = -\left(\overline{a} * \overline{b} * \overline{c} + \overline{a} * \overline{b} * \overline{d} + \overline{b} * \overline{c} * \overline{d} + \overline{a} * \overline{c} * \overline{d}\right)$$
$$\overline{q} = \overline{a} * \overline{b} + \overline{b} * \overline{c} + \overline{c} * \overline{a} + \overline{a} * \overline{d} + \overline{b} * \overline{d} + \overline{c} * \overline{d}$$
$$\overline{p} = -\left(\overline{a} + \overline{b} + \overline{c} + \overline{d}\right).$$

Using the known expansions of the functions, the coefficients are listed in Table A6.1 below as convolution numbers of length $n = 12$.

Table A6.1.

i	$\overline{s}$	$\overline{r}$	$\overline{q}$	$\overline{p}$
0	2	−7	9	−5
1	−2	4.5	−3	.5
2	.25	−.375	.5	−.375
3	−.1666667	1.9375	−2.5	.7291667
4	.3177084	−2.013021	2.447917	−1.002604
5	−.3614584	1.916406	−2.511458	.8940104
6	.3252170	−1.670204	2.312674	−.9781033
7	−.3491568	1.763388	−2.474033	.9621450
8	.3286445	−1.627989	2.337015	−.9869328
9	−.3464532	1.702059	−2.452751	.9890406
10	.3309496	−1.612433	2.351416	−.9907262
11	−.3445821	1.673126	−2.438432	.9991457
12	.3324845	−1.609423	2.361894	−.9929922
l	0	0	0	
m	12	12	12	12
p	12	12	12	12

The solution method to find $\overline{u}$ from equation (A6.2) starts by determining u_0 which according to Theorem 3 is the root of the scalar polynomial

$$f_0(u_0) = s_0 + r_0 u_0 + q_0 u_0^2 + p_0 u_0^3 + u_0^4 = 0, \tag{A6.3}$$

where the coefficients are the leading terms from Table A6.1. A polynomial root solver supplies the result

$$u_0 = 2, \quad \text{single root,}$$
$$u_0 = 1, \quad \text{repeated root, three times.}$$

The convolution derivative is

$$\overline{g} \equiv d\overline{f}/d\overline{u} = \overline{r} + 2\overline{q} * \overline{u} + 3\overline{p} * \overline{u}^{*2} + \overline{u}^{*3}. \tag{A6.4}$$

For the single root u_0 we obtain the leading element g_0 from

$$g_0 = r_0 + 2q_0 u_0 + 3p_0 u_0^2 + u_0^3 = 1 \neq 0,$$

therefore, the advance pointer $a = 1$. Set the convolution number

$$\overline{\delta}_g = g_0 \overline{\delta}_0 = \{g_0, 0, \ldots\}, \quad l = m = 0, \quad p = n.$$

We compute $\overline{f}$ by Horner's scheme

$$\overline{f} = \overline{s} + \overline{u} * \Big(\overline{r} + \overline{u} * \Big(\overline{q} + \overline{u} * \underbrace{\Big(\underbrace{\overline{p} + \overline{u}}_{\overline{v}_1}\Big)}_{\overline{v}_2}\Big)\Big)$$

$\overline{v}_3$

$\overline{v}_6$

with the indicated intermediate variables $\overline{v}_1 \ldots \overline{v}_6$ as shown. The advance pointer program is:

Set virgin pointer convolution numbers $\overline{v}_1, \ldots \overline{v}_6, \overline{f}$

set $\overline{u} = \{u_0, 0, \ldots\}, \quad l = m = p = 0$

$$
\begin{aligned}
&\text{for } i = 1 \text{ to } n \\
&\overline{v}_1 = \overline{p} + \overline{u} \\
&\overline{v}_2 = \overline{u} * \overline{v}_1 \\
&\overline{v}_3 = \overline{q} + \overline{v}_2 \\
&\overline{v}_4 = \overline{u} * \overline{v}_3 \\
&\overline{v}_5 = \overline{r} + \overline{v}_4 \\
&\overline{v}_6 = \overline{u} * \overline{v}_5 \\
&\overline{f} = \overline{s} + \overline{v}_6 \\
&\overline{u} = -\overline{f} / \overline{\delta}_g \\
&\text{advance pointer in } \overline{u} : \quad p = p + a \\
&\text{next } i
\end{aligned}
\tag{A6.5}
$$

The result is shown in the first column of Table A6.2, and the exact solution $\overline{a}$ in the next column. Some numerical error appears in the higher elements, which is due to digital number truncation, since according to Theorem 4 the elements should be exact, noting that element u_0 was exact. The application of Theorem 4 to the solution of nonlinear equations is limited because generally the root of the scalar function $f(u_0)$ cannot be determined exactly.

For the solution with the repeated root $u_0 = 1$, we may expect at most three solutions. Therefore, we set the advance pointer $a = 3$ and check the compatiblity of f_1 and f_2, according to equation (81), by computing $\overline{f}$ with the same equations (A6.5) above, using advance pointer numbers, with the result $\overline{f} = \{0, 0, 0, 1.788139E-07 \ldots\}$. The elements f_1 and f_2 are indeed zero and, therefore, solutions exist. The fact that f_3 is also ≈ 0 is just by chance because $u_1 = 0$ is in one of the possible solutions.

According to Sections 11 and 15 the solution of u_1 is obtained from the three possible roots of $f_3(u_0, 0, 0)$ for which we choose the method of writing f_3 explicitly as polynomial in u_1

$$f_3 = \alpha_0 + \alpha_1 u_1 + \alpha_2 u_1^2 + \alpha_3 u_1^3,$$

where according to Section 15 $\alpha_0 = f_{xxx0}$, $\alpha_1 = 3g_{xx0}$, $\alpha_2 = 3h_{x0}$, $\alpha_3 = l_0$, . The two additional convolution derivatives are

$$\overline{h} \equiv d\overline{g}/d\overline{u} = 2\overline{q} + 6\overline{p} * \overline{u} + 12\overline{u}^{*2}, \tag{A6.6}$$

$$\overline{l} \equiv d\overline{h}/d\overline{u} = 6\overline{p} + 24\overline{u}. \tag{A6.7}$$

Accordingly, the equations for the partial derivatives are

$$
\begin{aligned}
\overline{f}_{xxx} &= \overline{s}''' + \overline{r}''' * \overline{u} + \overline{q}''' * \overline{u}^{*2} + \overline{p}''' * \overline{u}^{*3}, \\
\overline{g}_{xx} &= \overline{r}'' + 2\overline{q}'' * \overline{u} + 3\overline{p}'' * \overline{u}^{*2}, \\
\overline{h}_x &= 2\overline{q}' + 6\overline{p}' * \overline{u}.
\end{aligned}
$$

Each one of these convolution numbers must be evaluated for u_0 only and, therefore, we need only the leading elements of all convolution coefficients in the equation above, which are rather obtained directly as

$$
\begin{aligned}
&p_0' = p_1, \quad p_0'' = 2p_2, \quad p_0''' = 6p_3, \\
&q_0' = q_1, \quad q_0'' = 2q_2, \quad q_0''' = 6q_3, \\
&r_0'' = 2r_2, \quad r_0''' = 6r_3, \\
&s_0''' = 6s_3.
\end{aligned}
$$

The elements p_i, q_i, r_i, s_i are taken from the convolution constants in Table A6.1, from which the coefficients α_i are computed

$$\begin{aligned}\alpha_0 &= 6s_3 + 6r_3u_0 + 6q_3u_0^2 + 6p_3u_0^3 &&= 1.072884E-06,\\ \alpha_1 &= 3(2r_2 + 4q_2u_0 + 6p_2u_0^2) &&= -3,\\ \alpha_2 &= 3(2q_1 + 6p_1u_0) &&= -9,\\ \alpha_3 &= 6p_0 + 24u_0 &&= -6.\end{aligned}$$

With a polynomial rootsolver we find three distinct roots u_1 as listed below compared to the exact roots.

roots u_1		g_2	
computed	exact	computed	exact
3.56727E-07	0	−0.5000011	−0.5
−.5000007	−0.5	0.2500001	0.25
−.9999996	−1	−0.4999988	−0.5

Also listed are the corresponding values of g_2 which are computed in the program below. The errors are due to the error in α_0 which should be zero.

With these results the program for each root u_1 is as follows. We will have to program $\overline{g}$ to find g_2, which we cannot determine by a simple scalar equation like g_0 above.

We compute $\overline{g}$ by Horner's scheme,

$$\overline{g} = \overline{r} + \overline{u} * \overline{t} + \overline{u} * \Big(\overbrace{\overline{k} + \underbrace{4\overline{u}}_{\overline{v}_1}}^{\overline{v}_2} \Big)^{\overline{v}_3 \cdots}_{\overline{v}_5}$$

where the convolution constants $\overline{t} = 2\overline{q}$, $\overline{k} = 3\overline{p}$, and using the same intermediate variables $\overline{v}_1 \ldots \overline{v}_5$ as shown, for which we may as well use advance pointer numbers. The result will also confirm for each root that g_0 and g_1 are indeed zero. If we should also find $g_2 = 0$ we would have to continue the program quite differently as indicated in Section 15.

The advance pointer program is, with advance pointer $a = 3$,

Set virgin pointer convolution numbers $\overline{v}_1, \ldots \overline{v}_5, \overline{g}$

set $\overline{u} = \{u_0, u_1, \ldots\}$, $l = 0, m = p = 1$

$$\begin{aligned}\overline{v}_1 &= 4\overline{u}\\ \overline{v}_2 &= \overline{k} + \overline{v}_1\\ \overline{v}_3 &= \overline{u} * \overline{v}_2\\ \overline{v}_4 &= \overline{t} + \overline{v}_3\\ \overline{v}_5 &= \overline{u} * \overline{v}_4\\ \overline{g} &= \overline{r} + \overline{v}_5.\end{aligned} \qquad \text{(A6.8)}$$

Set $\overline{\delta}_g = g_2\overline{\delta}_2 = \{0, 0, g_2, 0, \ldots\}$, $l = m = 2$, $p = n$.

Reset virgin pointer convolution numbers $\overline{v}_1, \ldots \overline{v}_6, \overline{f}$

for $i = 1$ to n

follow equations (A6.5)

next i

The results are shown in Table A6.2 with the exact solution for comparison. Note that the last three convolution roots can only be determined to length $n-2$.

The tables convey an impression of the accuracy that can be expected. The greatest accuracy occurs at the second solution, which seems to be related to the fact that the theoretical radius of convergence is infinite, while of the first root it is $\frac{\pi}{2}$, and the two last it is one. In all cases, the numerical radius of convergence is simply so much less than the theoretical and cannot be increased by taking larger convolution number length, because it is due to the digital truncation error of the numbers. It is also obvious that the roots found from the nonlinear equation are much less accurate than the convolution numbers from computing the theoretical coefficients of the functions, while according to Theorem 4 they should both be exact. This is not due to the initial error from the polynomial root solving routine for u_0 and then u_1 as a new calculation with exact leading elements values will show. It is rather due to the increased digital truncation error that occurs when each root must be filtered out as it were from a numerical combination of all four roots. Note also that there is an apparent similarity between this method of solution and the one in Section 17.1 inasmuch both require the programming of the convolution derivative $\overline{g}$. However, first of all, in the method here the additional computation of dividing by a full convolution number is avoided, but secondly, the method of Section 17.1 cannot be applied although the division is compatible. The reason is that the pointer is not advanced in the loop, and checking on the details reveals that the next unknown element occurs on both sides of the equation.

We have used the convolution equation (A6.2) as the governing equation to determine the roots, but this equation is limited to the smallest radius of convergence of the factors $\overline{a}$, $\overline{b}$, $\overline{a}$, $\overline{a}$ which is one, and, therefore, no Taylor expansion can be found in a region $x > 1$. A true definition of the governing equation is equation (A6.1), where the first two functions must are defined by some particular equations. Then the Taylor expansions for other regions beyond the singularities can be found by determining the appropriate convolution numbers $\overline{a}$, $\overline{b}$, $\overline{a}$, $\overline{a}$ which is one, for those regions.

Table A6.2.

i	$\overline{u_a}$	$\overline{a}$	$\overline{u_b}$	$\overline{b}$
0	2	2	1	1
1	1	1	3.576274E-07	0
2	0	0	−.4999995	−.5
3	.333333	.3333333	3.576271E-07	0
4	8.046627E-07	0	4.166653E-02	4.166667E-02
5	.1333307	.1333333	0	0
6	8.523464E-06	0	−1.389023E-03	−1.388889E-03
7	.0539436	5.396825E-02	2.980225E-07	0
8	6.949902E-05	0	2.503389E-05	2.480159E-05
9	.0216786	2.186949E-02	−1.192090E-07	0
10	5.158484E-04	0	−1.251695E-06	−2.755732E-07
11	7.477939E-03	8.863235E-03	0	0
12	3.701717E-03	0	0	2.087676E-09
l	0	0	0	0
m	12	11	10	12
n	12	12	10	12

Table A6.2 (cont.)

i	u_c	a	u_d	b
0	1	1	1	1
1	−.5000007	−.5	−.9999996	−1
2	−.1250008	−.125	.9999999	1
3	−6.250188E-02	−.0625	−.9999992	−1
4	−3.906379E-02	−.0390625	.9999954	1
5	−2.734743E-02	−2.734375E-02	−.9999957	−1
6	−2.050971E-02	−2.050781E-02	.9999721	1
7	−1.612388E-02	−1.611328E-02	−.9999562	−1
8	−1.310014E-02	−1.309204E-02	.9998329	1
9	−1.093971E-02	−1.091003E-02	−.9996404	−1
10	−9.294267E-03	−9.273529E-03	.9989045	1
11	0	−8.008957E-03	0	−1
12	0	−7.007837E-03	0	1
l	0	0	0	0
m	10	12	10	12
p	10	12	10	12

This numerical example is merely to illustrate the principle and should not convey the impression that large equations, for which the convolution number concept is intended, can be handled in the same way. First of all, a simple single letter notation for the coefficients has been used; in large systems coefficient names must be designed systematically with multiple symbols. Next, the analytic work was simple and done by hand; in large systems this may be complicated and will have to be done by symbolic computer.

We may also use this example to show how the role of the symbolic computer is changed with the concept of the convolution number. Previously, the program above could be implemented by a symbolic computer, then obtaining the Taylor coefficients analytically. An addition to the symbolic program would then translate the generally long analytical expressions to a digital programming language, e.g., using the feature of MACSYMA to translate into FORTRAN, [4], to determine the Taylor coefficients. With the present method, the symbolic computer may be used to develop the convolution solution routine and translate it into a digital programming language, similar to the program above.

APPENDIX 7
BLASIUS DIFFERENTIAL EQUATION

It can be shown that the Blasius solution $|u(z)|$ has the 120° rotational symmetry as evident in Figure 15(c), within the first radius of convergence, by substituting the value of $ze^{\hat{\imath}\pi/3}$ in the Taylor series about the point $z = 0$. Because only every third coefficient is nonzero, the result is

$$u\left(ze^{\hat{\imath}\frac{\pi}{3}}\right) = e^{-\hat{\imath}\frac{2\pi}{3}}u(z).$$

The previous discussion illustrates the importance of the theoretical radius of convergence. The Blasius solution has a reported singularity at $x = -5.732308$, [60], converted to the form of equation (127). The number of 4.05335 by Shanks (converted) is therefore clearly in error, as suspected by [59], it is even less than the numerical radius of 4.5 which was achieved with the HP 9836 computer. We recalculated the position of the singularity on the negative real axis at −5.69003805, which means that two are straddling the positive real axis at $z = 5.69003805e^{\pm\hat{\imath}\pi/3}$, by analytical continuation as follows.

Function values close to a singular point cannot be evaluated using a Taylor expansion about a point too far away because the numerical radius of convergence is always considerably smaller than the theoretical. The singular point can be approached by analytic continuation in ever decreasing intervals equal to the ever decreasing theoretical and therefore numerical radius of convergence. By this method, not only the function values but also the position of the singular point is found within a small distance of the order of the truncation error of the digital numbers used. A fraction of this distance is the last numerical radius of convergence and value of x whose powers are used in the Taylor expansion to express the large value of the function or its derivatives. These two factors cause the corresponding Taylor coefficients to be very large preventing numerical computation. To counteract this, the variable x is transformed to variable v as soon as the radius of convergence r becomes small, so that the analytic function u is transformed to a function g. The transformations are

$$\begin{aligned} v &= \tfrac{x}{r} \\ u(x) &= g(v) = g\left(\tfrac{x}{r}\right) \\ u' &= g'\frac{1}{r} \\ u'' &= g''\frac{1}{r^2} \\ u''' &= g'''\frac{1}{r^2}, \end{aligned}$$

where $u' = \frac{du}{dx}$ and $g' = \frac{dg}{dv}$. Transform also to the new function

$$h = g\,r,$$

so that the new equation becomes

$$\begin{aligned} hh'' + 2h''' &= 0 \\ u &= h\frac{1}{r} \\ u' &= h'\frac{1}{r^2} \\ u'' &= h''\frac{1}{r^3} \\ u''' &= h'''\frac{1}{r^4}. \end{aligned}$$

This transformation is repeated in each interval which is smaller than the previous one, resulting in a well conditioned computation with an accumulative increasing transformation factor. The singular point can be approached with any chosen convolution number length n. With this method we found the singularity at $x = -5.690037$ with $n = 10$ in 142 intervals, at $x = -5.690039$ with $n = 40$ in 43 intervals on a IBM PC using 7 digits, and at $x = -5.6900380546$ with $n = 50$ in 67 intervals on the HP 9836.

Reference [60] used the largest interval straddled by the three values x_k computed from

$$\begin{aligned} u(x) &= \frac{3}{x - x_1}, \\ u'(x) &= \frac{-3}{(x - x_2)^2}, \\ u''(x) &= \frac{6}{(x - x_3)^3}, \end{aligned}$$

to determine bounds for the position of the singularity from the function values at x on the negative axis. We found that this was only an approximation, the interval only contained the singular point when the singular point was already reached.

The singularity at this point is not a pole, the function has a branchpoint there with a discontinuity which we computed by analytic continuation from the origin in the complex plane, once in the top half plane and once in the lower half plane, to a point $x = -6.66$, along a path which is always within the radius of convergence of the singularity.

To attempt an asymptotic solution in terms of Taylor series, valid at the origin and a good approximation at ∞, the form

$$u = x + \frac{1}{p(x)} - \frac{1}{p_0}$$

was tried. The Taylor transform is

$$\overline{u} = \overline{\delta}_1 + 1/\overline{p} - 1/p_0,$$

therefore, $\overline{p}$ was computed by convolution algebra from the previously computed solution $\overline{u}$ of the expansion about the origin

$$\overline{p} = \frac{1}{\overline{u} + 1/p_0 - \overline{\delta}_1}.$$

The value of $1/p_0 = 0.2792123425$ was taken from the asymptotic value of the previous solution. For $n = 40$ the result is a very good approximation for all positive values of x. The nearest three zeros of $p(x)$, which now approximate the singularities of the solution, were found at

$$\begin{aligned} &-5.637247 \\ &5.6988\angle + 61.965° \\ &5.6988\angle - 61.965° \end{aligned}$$

which are indeed closely the nearest singularities of the solution.

We have not pursued this method any further, we could not find any rational method to determine the constant p_0 other than from a previous solution. Also, the solution of the differential equation for $p(x)$ that is produced when the asymptotic form is substituted in the differential equation for $u(x)$, obtained by pointer convolution numbers, is not nearly as accurate as the algebraic solution.

APPENDIX 8
REVERSION OF POLYNOMIAL, EXAMPLE 4

Numerical details of the polynomial in Section 20 are given here for reference of typical results. The coefficients of the eighth degree polynomial are

$$\begin{aligned} w_0 &= 4 && + \hat{\imath}\,12 \\ w_1 &= 5.915232 && - \hat{\imath}\,5.779296 \\ w_2 &= -12.83945 && + \hat{\imath}\,14.12717 \\ w_3 &= -5.03268 && + \hat{\imath}\,1.072631 \\ w_4 &= -4.93746 && + \hat{\imath}\,.7171197 \\ w_5 &= -2.18088 && - \hat{\imath}\,2.58336 \\ w_6 &= -.129 && - \hat{\imath}\,1.253 \\ w_7 &= .3407143 && - \hat{\imath}\,.3664286 \\ w_8 &= .09375 && + \hat{\imath}\,.01875. \end{aligned}$$

The roots of the scalar equations appear as leading element in the eight convolution roots (pointers not shown):

Table A8.

	plane 1		plane 2	
i	real part	imag. part	real part	imag. part
0	1.130966	2.468194	−1.968876	−1.672515
1	2.892615E-03	−3.610062E-03	−2.331191E-03	−1.346490E-04
2	4.105577E-05	6.776909E-06	8.801477E-06	−5.947552E-06
3	2.125534E-07	5.330633E-07	−2.556894E-08	7.087508E-08
4	−5.611464E-09	7.563128E-09	−1.750968E-10	−6.147711E-10
5	−1.694708E-10	−1.372859E-11	4.677618E-12	3.724869E-12
6	−1.503084E-12	−2.894176E-12	−5.940899E-14	−3.160458E-15
7	3.300588E-14	−5.625751E-14	5.215331E-16	−3.308532E-16
8	1.344467E-15	−4.237650E-17	−2.499769E-18	6.123742E-18
9	1.589852E-17	2.353142E-17	−1.653458E-20	−7.062552E-20
10	−2.454295E-19	5.592800E-19	6.074171E-22	5.365723E-22
11	−1.318427E-20	1.990735E-21	−9.141810E-24	−9.770258E-25
12	−1.928830E-22	−2.228340E-22	9.214197E-26	−5.159637E-26

	plane 3		plane 4	
i	real part	imag. part	real part	imag. part
0	−.1916872	−1.510858	−1.751260	2.369521
1	−1.351728E-02	−2.761571E-03	−4.594956E-03	−4.782129E-03
2	1.341326E-04	−4.553166E-04	−7.925985E-05	1.468293E-06
3	2.413744E-05	1.035080E-05	−1.037830E-06	8.859213E-07
4	−8.962914E-07	1.519583E-06	−6.782880E-09	2.644659E-08
5	−1.036730E-07	−8.109610E-08	1.832536E-10	5.701394E-10
6	7.492548E-09	−7.335027E-09	1.045098E-11	9.413013E-12
7	5.230074E-10	6.996511E-10	3.399880E-13	7.933824E-14
8	−6.566035E-11	3.661496E-11	8.756468E-15	−2.350988E-15
9	−2.430622E-12	−6.170555E-12	1.788119E-16	−1.661720E-16
10	5.791714E-13	−1.427434E-13	2.230253E-18	−6.380123E-18
11	5.871223E-15	5.417874E-14	−2.870977E-20	−1.890094E-19
12	−5.041459E-15	−1.302314E-16	−3.281239E-21	−4.410103E-21

	plane 5		plane 6	
i	real part	imag. part	real part	imag. part
0	−2.361486	1.856237	1.800305	1.114940
1	5.116708E-03	−1.669765E-03	1.601835E-03	−8.685653E-03
2	4.943098E-05	−3.555053E-05	1.643081E-04	3.677668E-05
3	5.476037E-07	−9.115067E-07	−1.129469E-06	4.797425E-06
4	2.125759E-09	−2.213376E-08	−1.676511E-07	−3.898479E-08
5	−1.855077E-10	−4.797869E-10	1.436843E-09	−6.471883E-09
6	−9.495169E-12	−8.434570E-12	2.661255E-10	5.515892E-11
7	−3.183971E-13	−7.937101E-14	−2.172378E-12	1.142980E-11
8	−8.511615E-15	2.098745E-15	−5.067048E-13	−8.683293E-14
9	−1.789786E-16	1.603116E-16	3.491421E-15	−2.300988E-14
10	−2.303674E-18	6.312660E-18	1.064749E-15	1.400088E-16
11	2.698774E-20	1.891059E-19	−5.544392E-18	5.001916E-17
12	3.261330E-21	4.432948E-21	−2.378974E-18	−2.138998E-19

Table A8 (cont.).

	plane 7		plane 8	
i	real part	imag. part	real part	imag. part
0	−.1612757	−.6957161	.7604568	.5273415
1	3.838503E-02	5.144643E-04	−2.755275E-02	2.112938E-02
2	−2.979022E-05	−1.06651E-04	−2.886787E-04	5.584445E-04
3	−4.469071E-05	−2.315915E-05	2.198600E-05	7.432533E-06
4	−9.416685E-08	−7.778747E-07	1.168558E-06	−7.139867E-07
5	1.219595E-07	1.491957E-07	−1.955631E-08	−6.170780E-08
6	−3.222597E-09	7.848526E-09	−4.535509E-09	−5.667344E-10
7	−3.906866E-10	−9.403042E-10	−1.302032E-10	2.292771E-10
8	5.419720E-11	−5.066014E-11	1.196862E-11	1.413236E-11
9	1.258411E-12	6.534101E-12	1.168695E-12	−3.405254E-13
10	−5.862233E-13	2.271791E-13	5.984334E-15	−8.457782E-14
11	−5.653558E-16	−5.150123E-14	−5.300141E-15	−2.727807E-15
12	5.359208E-15	−1.479604E-16	−3.153440E-16	2.784296E-16

APPENDIX 9
ARCCOS FROM REVERSION OF TAYLOR SERIES

The following table lists the coefficients of $\arccos(x)$ as found by the approximate series approximation method in Section 20, compared to the machine exact coefficients found by the exact method (pointers not shown).

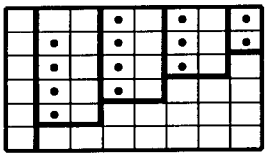
Table A9.

i	approx.	exact	i	approx.	exact
0	1.570796	1.570796			
1	−1	−1	16	4.144395E-08	0
2	8.940700E-08	0	17	−1.155185E-02	−1.15518E-02
3	−.1666668	−.1666667	18	3.921913E-08	0
4	7.450585E-08	0	19	−9.761656E-03	−9.761609E-03
5	−7.500008E-02	−.075	20	3.727630E-08	0
6	6.270911E-08	0	21	−8.390379E-03	−8.390335E-03
7	−4.464293E-02	−4.464286E-02	22	3.560205E-08	0
8	5.587942E-08	0	23	−7.312567E-03	−7.312526E-03
9	−3.038201E-02	−3.038194E-02	24	3.420683E-08	0
10	5.103656E-08	0	25	−6.447250E-03	−6.447210E-03
11	−2.237222E-02	−2.237216E-02	26	3.293672E-08	0
12	4.718710E-08	0	27	−5.740076E-03	−5.740038E-03
13	−1.735282E-02	−1.735276E-02	28	3.179816E-08	0
14	4.403835E-08	0	29	−5.153346E-03	−5.153310E-03
15	−1.396490E-02	−1.396484E-02	30	3.074932E-08	0

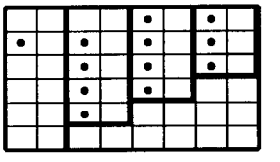
APPENDIX 10

Figure A10. Development of Transonic Nozzle Bi-Convolution Variables
$u(x,0) = 0.5x$

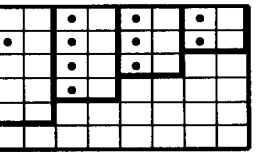
$L(\underline{\overline{\phi}}_{ly}) = \underline{\overline{\phi}}_y$



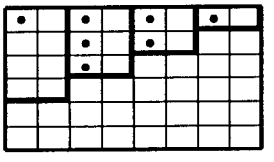
$\int \underline{\overline{\phi}}_y \partial y = \underline{\overline{\phi}}$



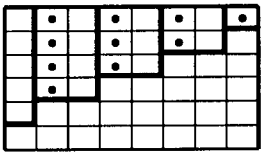
$\partial/dx\, \underline{\overline{\phi}} = \underline{\overline{\phi}}_x$



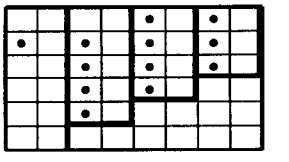
$\partial/dx\, \underline{\overline{\phi}}_x = \underline{\overline{\phi}}_{xx}$



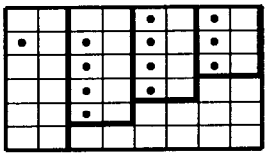
$\partial/dx\, \underline{\overline{\phi}}_y = \underline{\overline{\phi}}_{xy}$



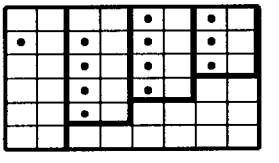
$\underline{\overline{\phi}}_x^{*2} = \underline{\overline{U}}s$



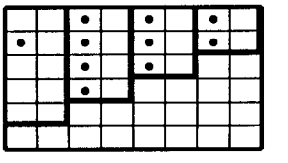
$\frac{\gamma+1}{2}\underline{\overline{U}}s = \underline{\overline{U}}a$



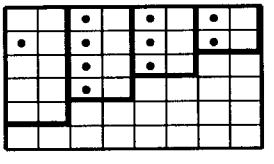
$\frac{\gamma-1}{2}\underline{\overline{U}}s = \underline{\overline{U}}b$



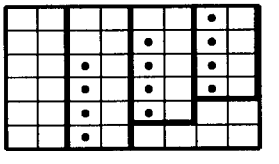
$(\gamma+1)\underline{\overline{\phi}}_x = \underline{\overline{U}}c$



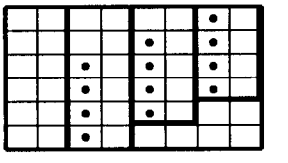
$(\gamma-1)\underline{\overline{\phi}}_x = \underline{\overline{U}}d$



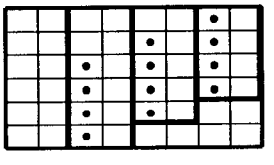
$\underline{\overline{\phi}}_y^{*2} = \underline{\overline{V}}s$



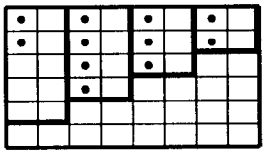
$\frac{\gamma+1}{2}\underline{\overline{V}}s = \underline{\overline{V}}a$



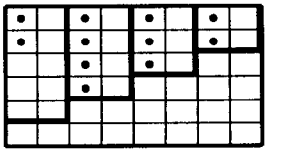
$\frac{\gamma-1}{2}\underline{\overline{V}}s = \underline{\overline{V}}b$



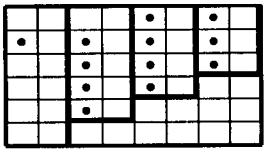
$1 + \underline{\overline{\phi}}_x = \underline{\overline{Q}}a$



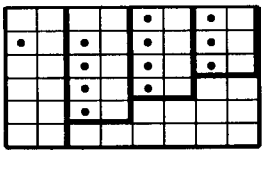
$2\underline{\overline{Q}}a = \underline{\overline{Q}}b$



$\underline{\overline{U}}a + \underline{\overline{V}}b = \underline{\overline{R}}$



$\underline{\overline{U}}b + \underline{\overline{V}}a = \underline{\overline{S}}$



$\underline{\overline{U}}b + \underline{\overline{V}}b = \underline{\overline{T}}$

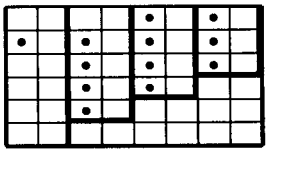
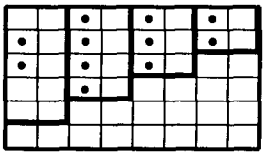
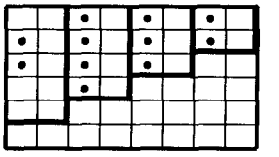


Figure A10. Development of Transonic Nozzle Bi-Convolution Variables (contd.)
$u(x,0) = 0.5x$

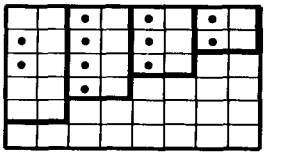
$\overline{R} + \underline{U}c = \underline{A}$



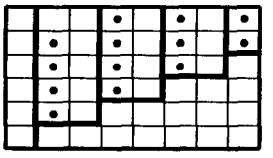
$-\overline{S} - \underline{U}d = \underline{B}1$



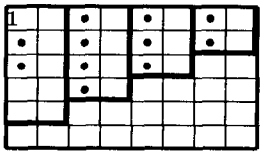
$-\overline{T} - \underline{U}d = \underline{D}1$



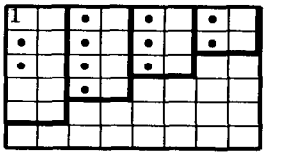
$\overline{Q}b * \underline{\phi}_y = \underline{C}$



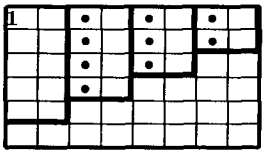
$1 + \overline{B}1 = \underline{B}$



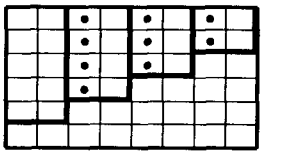
$1 + \overline{D}1 = \underline{D}$



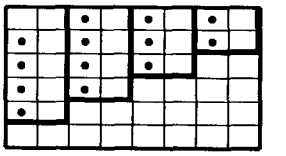
$\overline{D} \neq \overline{B} = \underline{E}$



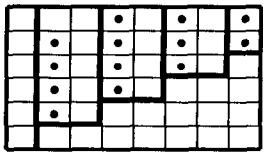
$\overline{E} - \overline{e}_1 = \underline{E}1$



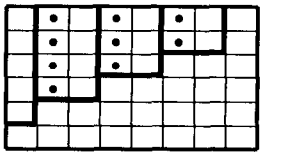
$\overline{A} \neq \overline{B} = \underline{F}$



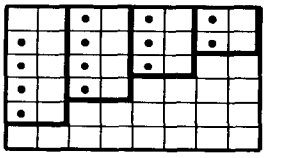
$\overline{C} \neq \overline{B} = \underline{G}\phi$



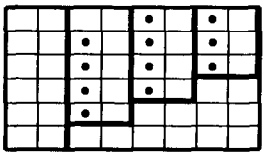
$\overline{E}1 \neq \overline{Y} = \underline{P}$



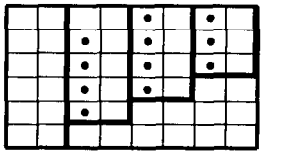
$\overline{F} * \underline{\phi}_{xx} = \underline{F}\phi$



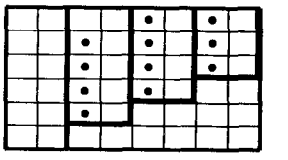
$\overline{G} * \underline{\phi}_{xy} = \underline{G}\phi$



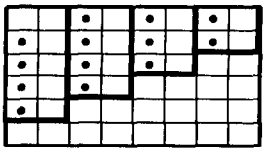
$\overline{P} * \underline{\phi}_y = \underline{P}\phi$



$\overline{G}\phi - \overline{P}\phi = \underline{H}\phi$



$\overline{F}\phi + \overline{H}\phi = \underline{\phi}_{ly}$



REFERENCES

1. E.D. Rainville, *Special Functions*, Chelsea, New York, (1960).
2. M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, (1972).
3. *MAPLE Reference Manual*, 5th ed., Symbolic Computation Group, Waterloo, (1988).
4. Symbolics, In *MACSYMA User's Guide*, Symbolics, Inc., (1987).
5. P.F. Byrd and M.D. Friedman, *Handbook of Elliptic Integrals for Engineers and Scientists*, 2nd ed., Springer Verlag, Berlin, (1971).
6. R. Bulirsch, Numerical calculation of elliptic integrals and elliptic functions, *Numerische Mathematik* **7**, 78–90 (1965).
7. K. Knopp, *Theorie und Anwendung der Unendlichen Reihen*, Springer Verlag, Berlin/Göttingen/Heidelberg, (1964), English translation: *Theory and Applications of Infinite Series*, Blackie & Son, London (1928).
8. P. Henrici, Applied and computational complex analysis, Vol. 1, In *Power Series—Integration—Conformal Mapping—Location of Zeros*, John Wiley, New York, (1974).
9. R. Bracewell, *The Fourier Transform and its Applications* 2nd ed., McGraw Hill, New York, (1978).
10. S. Fenyö and T. Frey, *Moderne Methoden in der Technik*, Birkhäuser Verlag, Basel, (1967).
11. J. Mikusiński, *Operational Calculus*, 5th ed., London, (1958).
12. R. Courant, *Vorlesungen über Differential und Integralrechnung*, Springer-Verlag, Berlin, (1955); English translation: *Differential and Integral Calculus*, 2nd ed., Blackie & Son, London (1937).
13. G.A. Gibson, *Advanced Calculus*, Macmillan, London, (1958).
14. E.W. Ng, Symbolic-numeric interface: A review, In *Lecture Notes in Computer Science* **72**, *Symbolic and Algebraic Computation*, (Edited by E.W. Ng), pp. 330–345, Springer-Verlag, Berlin/Göttingen/Heidelberg, (1979).
15. R.P. Brent and H.T. Kung, $O((n \log n)^{3/2}$ *Algorithms for Composition and Reversion of Power Series*, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, (1975).
16. R.P. Brent and H.T. Kung, *Fast Algorithms for Manipulating Formal Power Series*, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, (1976).
17. R.T. Moenck, Practical fast polynomial multiplication, *Symbolic and Algebraic Computation*, (Edited by E.W. Ng), Lecture Notes in Computer Science 72, pp. 136–148, Springer-Verlag, Berlin, (1979).
18. G. Adomian, Convergent series solution of nonlinear equations, *Journal of Computational and Applied Mathematics* **11**, 225–230 (1984).
19. G. Adomian, Solution of the Navier-Stokes equation–I. Partial differential equation systems, *Computers Math. Applic.* **12A** (11), 1119–1124 (1986).
20. G. Adomian and R. Rach, A general approach to solution of partial differential equation systems, *Computers Math. Applic.* **13** (9–11), 741–747 (1987).
21. G. Adomian and R. Rach, Transformation of series, *App. Math. Lett.* **4** (4), 73–76 (1991).
22. G. Adomian, A review of the decomposition method and some recent results for nonlinear equations, *Computers Math. Applic.* **21** (5), 101–127 (1991).
23. G. Adomian and R. Rach, Generalization of Adomian polynomials to functions of several variables, *Computers Math. Applic.* **24** (5/6), 11–24 (1992).
24. O. Rand, Harmonic variables—A new approach to nonlinear periodic problems, *Computers Math. Applic.* **15** (11), 953–961 (1988).
25. W.C. Hassenpflug, Matrix tensor notation, Part I, *Computers Math. Applic.* **4** (4), 73–76 (1992).
26. E.I. Jury, *Theory and Application of the z-Transform Method Repr.*, Krieger, New York, (1973).
27. D.E. Knuth, *The Art of Computer Programming Vol. 2*, Seminumerical Algorithms, Addison-Wesley, Reading, (1969).
28. J. Hagen, On division of series, *Am. Journ. of Math.* **5**, 236 (1883).
29. F.B. Hildebrand, *Advanced Calculus for Applications*, 2nd ed., Prentice Hall, Englewood Cliffs, (1976).
30. D.R. Hill, *Experiments in Computational Matrix Algebra,*, Random House, New York, (1988).
31. F.S. Tse, I.E. Morse and R.T. Hinkle, *Mechanical Vibrations, Theory and Practice*, Allyn and Bacon, Boston, (1963).
32. J.S. Przemieniecki, *Theory of Matrix Structural Analysis*, MacGraw Hill, New York, (1968).
33. L. Meirovitch, *Computational Methods in Structural Dynamics*, Sijthoff & Noordhoff, Alphen aan den Rijn, (1980).
34. R. Zurmühl, *Matrizen*, Springer-Verlag, Berlin, (1964).
35. R.E.D. Bishop, G.M.L. Gladwell and S. Michaelson, *The Matrix Analysis of Vibration*, Cambridge University Press, London, (1965).
36. W.C. Hassenpflug, Matrix tensor notation, Part II (to appear).
37. R.J. Guyan, Reduction of stiffness and mass matrices, *AIAA Journal* **3** (2), 380 (1965).
38. J.S. Archer, Consistent mass matrix for distributed mass systems, *Journal of the Structural Division, Proceedings of the American Society of Civil Engineers* **89** (ST 4), 161–178 (Aug. 1963).
39. R.D. Cook, *Concepts and Applications of Finite Element Analysis*, 2nd ed., John Wiley, New York, (1981).
40. L.M. Milne-Thomson, *Theoretical Hydrodynamics*, 5th ed., Macmillan, London, (1968).
41. W.C. Hassenpflug, Revised notation for partial derivatives, *Computers Math. Applic.* **26** (3), 95-106 (1993).
42. P. Henrici, *Applied and Computational Complex Analysis, Vol. 2*, John Wiley, New York, (1977).
43. D. Küchemann and J. Weber, *Aerodynamics of Propulsion*, McGraw Hill, New York, (1953).

44. D. Barton, I.M. Willers and R.V.M. Zahar, Taylor Series Methods for Ordinary Differential Equations—An Evaluation, In *Mathematical Software*, (Edited by J.R. Rice), Academic Press, New York, (1971).
45. G.A. Korn and T.M. Korn, *Electronic Analog Computers*, 2nd ed., McGraw Hill, New York, (1956).
46. F.H. Raven, *Automatic Control Engineering*, 2nd ed., McGraw Hill, New York, (1968).
47. W.T. Thomson, *Theory of Vibrations with Applications*, 2nd ed., Prentice Hall, Englewood Cliffs, (1981).
48. C. Hayashi, *Nonlinear Oscillations in Physical Systems*, McGraw Hill, New York, (1964).
49. B. Gold and C.M. Rader, *Digital Processing of Signals*, McGraw Hill, New York, (1969).
50. D.J. DeFatta, J.G. Lucas and W.S. Hodgkiss, *Digital Signal Processing: A Design Approach*, John Wiley, New York, (1988).
51. W. Giloi and R. Herschel, *Rechenanleitung für Analogrechner*, AEG Telefunken, Konstanz, (1961).
52. A. Sydow, *Programmierungstechnik für elektronische Analogrechner*, VEB Verlag Technik, Berlin, (1964).
53. M. Schuler and H. Gebelein, *Eight and Nine Place Tables of Elliptical Functions*, Springer-Verlag, Berlin, (1955).
54. C.R. Wylie Jr., *Advanced Engineering Mathematics*, 3rd ed., McGraw Hill, New York, (1966).
55. *IMSL, Math Library User's Manual*, IMSL, Houston, Texas, (1987).
56. V. Pereyra, PASVA3: An adaptive finite-difference FORTRAN program for first order nonlinear boundary value problems, In *Lecture Notes in Computer Science*, Vol. 76, 67–88, Springer Verlag, Berlin, (1978).
57. H. Schlichting, *Boundary Layer Theory* 7th ed., McGraw Hill, New York, (1979).
58. S. Goldstein, Ed., *Modern Developments in Fluid Dynamics, Vol. I*, Oxford University Press, (1938).
59. L. Rosenhead, *Laminar Boundary Layers*, Clarendon Press, Oxford, (1963).
60. J. Steinheuer, Die Lösungen der Blasiusschen Grenzschichtdifferentialgleichung, *Braunschweigsche Wissenschaftliche Gesellschaft Abhandlungen* **20**, 96–125 (1968).
61. F.M. White, Jr., B.F. Barfield and M.J. Goglia, Laminar flow in a uniformly porous channel, *Journal of Applied Mechanics* **25** (4), 613–617 (Dec. 1958).
62. F.M. White, Jr., Laminar flow in a uniformly porous tube, *Journal of Applied Mechanic* **29** (1), 201–204 (March 1962).
63. C.E. van Ostrand, Reversion of power series, *Philos. Mag.* **19** (6), 366 (1910).
64. G. Adomian and R. Rach, Nonlinear transformation of series—Part II, *Computers Math. Applic.* **23** (10), 79–83 (1992).
65. A.H. Shapiro, *The Dynamics and Thermodynamics of Compressible Fluid Flow, Vol. II*, Ronald Press, New York, (1954).
66. M.J. Zucrow and J.D. Hoffman, *Gas Dynamics, Vol. II*, John Wiley, New York, (1977).
67. K. Oswatitsch, *Gasdynamik*, Springer Verlag, Wien, (1952).
68. T. Meyer, Dissertation Göttingen, Forschungsheft VDI 62, (1908).
69. G.J. Taylor, The flow of air at high speeds past curved surfaces, *R & M* **1381** (1930).
70. R. Sauer, *General Characteristics of the Flow Through Nozzles at Near Critical Speeds, NACA TM 1147*, (1947).
71. K. Oswatitsch and W. Rothstein, *Flow Pattern in a Converging-Diverging Nozzle, NACA TM 1215*, (1949).
72. I.M. Hall, Transonic flow in two-Dimensional and axially-symmetric nozzles, *Quarterly Journal of Mechanics and Applied Mathematics* **XV**, Pt. 4, 487–508 (1962).
73. J.R. Kliegel and J.N. Levine, Transonic flow in small throat radius of curvature nozzles, *AIAA* **7** (7), 1375–1378 (1969).
74. M.J. Zucrow and J.D. Hoffman, *Gas Dynamics, Vol. I*, John Wiley, New York, (1976).
75. A.H. Nayfeh, *Introduction to Perturbation Techniques*, John Wiley, New York, (1981).
76. A.H. Nayfeh and D.T. Mook, *Nonlinear Oscillations*, John Wiley, New York, (1979).