COMPUTATIONAL PROBLEMS IN PRODUCING TAYLOR COEFFICIENTS FOR THE ROTATING DISK PROBLEM

J.A. van Hulzen

Twente University of Technology
Department of Applied Mathematics
P.O. Box 217, 7500 AE Enschede
The Netherlands

1. INTRODUCTION

During the past few years I had the pleasure to cooperate with some of my colleagues, D. Dijkstra, P. Gragert and P.J. Zandbergen, in studying an intriguing problem in fluid mechanics. My intention is not to produce polished results explaining the phenomena we examined - we plan to publish them elsewhere - but merely to describe the process we went through. Is it prototypical?
When glancing over the special issue of the SIGNUM Newsletter of February 1979, entitled "Numerical Computation, its nature and research directions", it stroke me that I could not find the letters S.A.M. or any similar indication. Perhaps only implicitly in some of the references. This contribution can be considered as one of the many possible reactions to Dick Jenks' PUSH 80 [12], by raising the following question: "What do we need when speaking about a symbolic-numeric interface: Exact or Reliable arithmetic?"

2. PROBLEM DESCRIPTION

During the last decade there has been a considerable interest in the study of solutions of the Navier-Stokes equations for the case of a rotating fluid above an infinite disk, both from a theoretical and a computational point of view, c.f. for instance [7,19].
Assume the disk is rotating with angular velocity Ω and the fluid with angular velocity at infinity $s\Omega$; the ratio s of these quantities is an essential parameter.
In a cylindric coordinate system (r,ϕ,z), when identifying the disk with the plane $z = 0$, the corresponding velocities are:

$$u = r\Omega f'(x),\ v = r\Omega g(x),\ w = -z(\nu\Omega)^{\frac{1}{2}} f(x)$$

where ν denotes the viscosity, $x = z(\Omega/\nu)^{\frac{1}{2}}$ and a prime denotes differentiation with respect to $x \in [0,\infty]$.
In terms of the functions f and g the Navier-Stokes equations reduce to the similarity form:

$$(2.1)\quad \begin{cases} f''' = f'^2 - g^2 - 2ff'' + s^2 \\ g'' \ = 2f'g - 2fg' \end{cases} .$$

This was shown by van Karman in 1921, c.f. Schlichting [15]. The boundary conditons are:

$$x = 0:\ f = 0,\ f' = 0,\ g = 1$$
$$x = \infty:\ f' = 0,\ g = s.$$

The reason that the structure of the solutions to these similarity equations for the rotating disk problem has received considerable attention is certainly the fact that, in the steady case, this set of ordinary differential equations still provides an exact representation of the original Navier-Stokes equations.

Introducing the complex velocity $\zeta = u + iv = r\Omega h$ and assuming an overbar denotes the complex conjugate, we can reformulate eq. (2.1) as follows:

$$h'' = h^2 - 2fh' + s^2$$
$$2f' = h + \bar{h}$$

Now the boundary conditions are:

$$x = 0:\ h = i,\ f = 0$$
$$x = \infty:\ h = is.$$

If the velocity at the disk is normalized to unity then the fluid at infinity is supposed to rotate with velocity s. If $|s| \to \infty$ it is preferable to use $\sigma = 1/s$. So, we obtain an alternative system by posing that the fluid is rotating with an angular velocity at infinity ω and at the disk with $\sigma\omega$:

$$(2.2)\begin{cases} H'' = H^2 - 2FH' + 1 \\ 2F' = H + \overline{H} \end{cases}$$

For $x = 0$ the boundary conditions are $H = i\sigma$, $F = 0$ and for $x = \infty$ we have $H = i$.
Now considering eq. (2.2) as a perturbation of the situation at $\sigma = 1$, i.e. taking the angular velocity at infinity as reference velocity ($\sigma = 1 + \delta$), we set

$$H = \sum_{\ell=0}^{\infty} H_\ell \delta^\ell .$$

Thus, we get instead of eq. (2.2):

$$(2.3)\begin{cases} H''_\ell - 2iH_\ell = Q_\ell = \sum\limits_{j=1}^{\ell-1} \{H_j H_{\ell-j} - 2F_\ell H'_{\ell-j}\}, \quad \ell \geq 2 \\ H''_1 - 2iH_1 = 0 \\ H_0 = i \\ 2F'_\ell = H_\ell + \overline{H}_\ell, \quad \ell \geq 1 . \end{cases}$$

Observe that

$$(2.4)\begin{cases} H_0 = i \\ H_1 = ie^{-(1+i)x} \\ 2F_1 = -\frac{1}{2}(1+i)e^{-(1+i)x} - \\ \quad -\frac{1}{2}(1-i)e^{-(1-i)x} + 1, \end{cases}$$

since for $x = 0$ the boundary conditions are $H_1 = i$, $H_\ell = 0$, $\ell \geq 2$ and $F_\ell = 0$, $\ell \geq 1$. For $x = \infty$ holds $H_\ell = 0$, $\ell \geq 1$.

Rogers and Lance [13] and Evans [8] provided numerical solutions. Bodonyi [1] made a significant contribution by demonstrating that the solution has an algebraic singularity at the critical value $\sigma = \sigma_{cr} = -0.69661$. However, none of the existing solutions was accurate enough.

The shearstress on the disk $H'(0)$ and the inflow $F(\infty)$ are interesting characteristic quantities of the phenomenon described by eq. (2.2). Instead of these quantities one can try to compute the coefficients of their Taylor expansions, c.f. eq. (2.3).

3. A FIRST APPROACH

The contributions of Dijkstra and Zandbergen to the solution of this one disk problem are significant [7,19]. Since they were also very interested in accurate solutions near the critical point $\sigma = \sigma_{cr}$, as to increase the understanding about the behaviour of the fluid, they asked me whether it might be profitable to apply the facilities of an algebra system on a variant of eq. (2.2). One can easily derive from these equations that, for $\ell \geq 2$, holds

$$H_\ell = \frac{1}{2+2i} \{\int_0^\infty e^{(1+i)(x+\xi)} Q_\ell(\xi)d\xi - \int_x^\infty (e^{(1+i)(x-\xi)} + e^{(1+i)(\xi-x)})Q_\ell(\xi)d\xi\}.$$

Eq. (2.4) then serves to start an iterative process. At a first glance, such a "simple" request seems to be an appropriate tool to extend the number of "believers in the gospel". Writing an adequate (FORMAC 73) program took less than an afternoon. Essentially one only needs to construct a subroutine capable to carry out the symbolic integration of elementary indefinite integrals of the form $\int x^k e^{\alpha x}dx$, where k is a positive integer and α is a Gaussian integer. But it turned out to be only a starting point of a lot of "trouble". Running the program on an IBM 360/50, in a partition of 390 K, it failed already for $\ell = 5$ after about 12 minutes CPU time, although the answers were correct and identical to earlier results of Rogers and Lance up to order $\ell = 3$ [13].

4. AN "IMPROVED" METHOD

4.1. An outline of the method and some results

Since storage and time requirements were too excessive, a more detailed analysis of the structure of the coefficients H_ℓ was necessary. As conjectured by Zandbergen, they ought to consist of finite sums of exponential functions, with predictable arguments, multiplied by polynomials of finite degree. We summarize the essentials of our analysis, confirming this conjecture [16].

Definition 4.1

Let $z_\ell(x)$, ℓ a positive integer, be an analytic function of the following structure:

$$z_\ell(x) = \sum_{k=1}^{\ell} \sum_{m=1}^{k+1} p_{\ell,k,m}\xi_{k,m},$$

where

$$\xi_{k,m} = \exp((a_k + ib_{k,m})x)$$

if

$$a_k = -k$$

and

$$b_{k,m} = m + (1-2m)\ \mathrm{rem}(k+m,2)^{*)},$$

and where

$$p_{\ell,k,m} = \sum_{n=0}^{\ell-k} \pi_{\ell,k,m,n} x^n$$

such that

$$\pi_{\ell,k,m,n} \in C. \qquad \square$$

Definition 4.1 describes in fact the structure of the coefficients H_ℓ as stated in:

Theorem 4.1

The $z_\ell(x)$, $\ell \geq 2$, of definition 4.1 satisfy eq. (2.3) if

1. $z_1(x) \equiv i\ \xi_{1,2}$
2. $\pi_{\ell;1,2,0} = -\sum_{k=2}^{\ell} \sum_{m=1}^{k+1} \pi_{\ell,k,m,0},\ \ell \geq 2.$ $\square$

Proof (a sketch):

Definition 4.1 is based on conjections concerning the structure of the solution, as indicated by the fundamental paper of Rogers and Lance [13], by linearizing the equations about the state of rigid rotation.
The exponentials $\xi_{k,m}$ are elements of a commutative, multiplicative monoid, generated by $\xi_{1,2} = e^{-(1+i)x}$ and $\xi_{1,1} = e^{-(1-i)x} = \overline{\xi}_{1,2}$. This is a natural consequence of the equations to be elaborated. Eq. (2.3) shows for $\ell = 2$ for instance that Q_2 is essential. Changing the (formal) name z_ℓ in H_ℓ, we have:

$$Q_2 = H_1^2 - 2F_1H_1'$$

where

$$H_1 = i\xi_{1,2},\ H_1' = (1-i)\xi_{1,2}$$

and

$$2F_1 = I(H_1(x)) + c_1 = -\frac{1}{2}\{(1+i)\xi_{1,2} + (1-i)\xi_{1,1}\} + 1$$

if

$$I(H_\ell(x)) = \int(H_\ell(x) + \overline{H}_\ell(x))dx \text{ and } c_\ell = -I(H_\ell(0)).$$

The products H_1^2 and $2F_1H_1'$ introduce a second set of exponentials:

$$\xi_{2,1} = e^{-2x} = \xi_{1,1}\xi_{1,2},\ \xi_{2,2} = e^{-(2-2i)x} = \xi_{1,1}^2$$

and

$$\xi_{2,3} = e^{-(2+2i)x} = \xi_{1,2}^2 = \overline{\xi}_{2,2}.$$

These exponentials are introduced in the description of H_2 via $Q_2 = H_2'' - 2iH_2$. This equation allows the construction of some recurrence relations between the coefficients of the polynomials, occuring in Q_2 and H_2. The degreebound for the polynomials $p_{\ell,k,m}$ of definition 4.1 is based on these relations and on the boundary conditions.
It is easily verified that

$$(4.2)\ \begin{cases} Q_2 = (i-1)\xi_{1,2} - i\ \xi_{2,1} \\ H_2 = \dfrac{1-2i}{10}\xi_{2,1} - \dfrac{1-2i+5x}{10}\xi_{1,2} \end{cases}$$

Hence some of the exponentials of the second set are missing in H_2, some others in $\overline{H}_2$, i.e. the corresponding polynomial coefficients are vanishing. These simple observations allow two essential conclusions:

1. The integration introduces rational arithmetic. A canonical representation of the Gaussian rationals requires "some" overhead in computing time but is necessary to avoid unwanted expression swell.
2. The boundary conditions, reflected by the second condition of theorem 4.1, demand for a summation of a number of Gaussian rationals. This number is a monotonic increasing function of ℓ. This can cause a coefficient length explosion; and it does indeed. Moreover, similar arguments hold for the integration constants c_ℓ.

*) rem(k+m,2) denotes the usual integer remainder when dividing the integer k + m by 2.

These conclusions are alarming, but can eventually be compensated by the apparant fact that some of the polynomials $p_{\ell,k,m}$ are vanishing.

Let us now complete the outline of the proof, which is constructive, i.e. it suggests how to design an algorithm to obtain the characteristic quantities. It is almost obvious that the structure of $z_\ell(x)$, $z_\ell'(x)$ and $I(z_\ell(x))$ is similar. Hence

$$(4.3)\begin{cases} z_\ell'(x) = \sum_{k=1}^{\ell} \sum_{m=1}^{k+1} r_{\ell,k,m}\xi_{k,m}, \text{ where} \\ r_{\ell,k,m} = \sum_{n=0}^{\ell-k} \rho_{\ell,k,m,n}\, x^n \end{cases}$$

and

$$(4.4)\begin{cases} I(z_\ell(x)) = \sum_{k=1}^{\ell}\sum_{m=1}^{k+1} u_{\ell,k,m}\xi_{k,m}, \text{ where} \\ u_{\ell,k,m} = \sum_{n=0}^{\ell-k} \mu_{\ell,k,m,n}\, x^n. \end{cases}$$

The quantities, which interest us are $H_\ell'(0)$ and $2F_\ell(\infty)$, or in the present terminology:

$$(4.5)\begin{cases} z_\ell'(0) = \sum_{k=1}^{\ell} \sum_{m=1}^{k+1} \rho_{\ell,k,m,0} \quad \text{and} \\ 2F_\ell(\infty) = -I(z_\ell(0)) = c_\ell = -\sum_{k=1}^{\ell} \sum_{m=1}^{k+1} \mu_{\ell,k,m,0} \end{cases}$$

Using the structure of z_ℓ, z_ℓ', $2F_\ell$ and c_ℓ we can construct

$$\overline{Q}_\ell(x) = \sum_{j=1}^{\ell-1} \{z_j z_{\ell-j} - z_j'\, 2F_{\ell-j}\}.$$

It requires a lot of elementary mathematics to "discover" that

$$(4.6)\begin{cases} \overline{Q}_\ell(x) = \sum_{k=1}^{\ell} \sum_{m=1}^{k+1} G_{\ell,k,m}\xi_{k,m}, \text{ where} \\ G_{\ell,k,m} = \sum_{n=0}^{\ell-k} \gamma_{\ell,k,m,n}\, x^n. \end{cases}$$

This analysis indicates that, for the construction of the $G_{\ell,k,m}$, it suffices to operate on the polynomials $p_{i,k,m}$, $r_{i,k,m}$, $u_{i,k,m}$ and the constants c_i; $i = 1,2, \ldots, \ell-1$. These $G_{\ell,k,m}$, or the $\overline{Q}_\ell(x)$, are required for the recurrence relations based on $\overline{Q}_\ell(x) = z_\ell'' - 2iz_\ell$ to obtain the next H_ℓ. The boundary conditions finally dictate the second condition of the theorem. □

The outline of the proof is given for two reasons:
1. To illustrate the type of operations required to produce the constants we are interested in $(H_\ell'(0),\ F_\ell(\infty))$.
2. To emphasize the inherent arithmetic problems related to attempts to solve this problem "error free". An important question is: Is it prototypical for a wider class of similar problems?" We return to this question in section 6.

Let us now discuss the first item. The above given indications suggest a simple algorithm to obtain the required Taylor expansions. The nature of the algorithm is iterative. Each step consists of two subalgorithms:
1. Determination of the polynomials $G_{\ell,k,m}$ performing elementary operations on polynomials with complex rational coefficients, such as addition and multiplication.
2. Determination of $H_\ell(x)$, with as a by-product the required constant $H_\ell'(0)$ and $F_\ell(\infty)$, operating on the coefficients of the $G_{\ell,k,m}$.

The advantages over the first approach are clear. Instead of operating on enormous expressions, representing the $H_\ell(x)$, we have only to deal with quite small univariate polynomials, stored in arrays, defining the same huge expressions.
A program can be written in two obvious ways:
1. Perform all operations on polynomials.
2. Perform all operations on coefficients, i.e. rational arithmetic suffices.

An ALTRAN-program based on the first possibility, delivered after hours of swopping, $H_\ell'(0)$, $F_\ell(\infty)$ for $\ell = 1,2, \ldots, 10$ and as given in table 1. Although we got 7 extra terms in comparison with Rogers and Lance [13], the expansion was disappointingly slowly converging. We simply had not yet enough information.
We observed another phenomenon, worthwhile mentioning. If $cg(\ell)$ denotes the coefficient growth as a function of ℓ, we roughly have $cg(\ell) = \ell(\ell-3)$. So $cg(11) = 88$, $cg(12) = 108, \ldots, cg(25) = 550, \ldots$ Again an improvement of the method seemed to be inevitable. But let us first try to get some insight in the weak aspects of the program.

ℓ	$C_\ell = 2 F_\ell(\infty)$	$C_\ell = 2 F_\ell(\infty)$	ℓ
10	827&5066826908&9351748079&2832917631&3658682236&9469997896&9452250507/ 262835&4988174167&8162184949&2001660572&6421309578&0005760000&000000000	0.0031483825	10
9	-3814650&8911873812&9131988594&8996308938&1709014973/ 756078643&7826196346&8148318332&9227379200&0000000000	- 0.0050483097	9
8	11665&8852693110&7017279779&7243126569/ 1528424&5897476647&0349556428&0000000000	0.0076326208	8
7	-8389&4296482041&6154919367/ 790971&8478538893&1200000000	- 0.0106064832	7
6	695380&6873580851/55592701&6944000000	0.0125084888	6
5	-136339103/1&5629120000	- 0.0087234024	5
4	-2987/221000	- 0.0135158371	4
3	7/80	0.0875000000	3
2	-3/10	- 0.3000000000	2
1	1	1.0000000000	1

T A B L E 1

ℓ	$H_\ell'(0)$	$H_\ell'(0)$			ℓ
1	- (I - 1)	1.0000000000	-	1.0000000000 i	1
2	- (2*I - 1) / 10	0.1000000000	-	0.2000000000 i	2
3	(9*I + 5) / 400	0.0125000000	+	0.0225000000 i	3
4	- (4185*I + 10246) / 884000	- 0.0115904977	-	0.0047341629 i	4
5	(13659287*I + 113590649) / 1&5629120000	0.0072678851	+	0.0008739639 i	5
6	(4987&8195098374*I - 240151&8749548181) / 55592701&6944000000	- 0.0043198454	+	0.0000897208 i	6
7	- 7 *(6&1493807271&6905947391*I - 57&6358048933&7225732525) / 158194&3695707778&6240000000	0.0025503476	-	0.0002721062 i	7
8	(58&9412430904&8575759864&0349533735*I - 369&7233676227&7593906291&9954922586) / 244547&9343596263&5255929028&4800000000	- 0.0015118646	+	0.0002410212 i	8
9	- (1018&7643431244&9061351505&7107801456&5657058591*I - 5452&7805330207&6016519785&2204846553&2246305103) / 6048629&1502609570&7745186546&6633819033&6000000000	0.0009014903	-	0.0001684290 i	9
10	(439305374&5090775641&2423941913&6536557012&0047141399&1775583976*I - 2270139102&129493759&6386590108&5961170458&1670347661&7964430545) / 420&5367981078&6685059495&9187202656&9162274095&3248009216&0000000000	- 0.0005398194	+	0.0004463005 i	10

4.2. Storage requirements

Assume we want to execute L steps of the algorithm. For some $\ell \le L$ the maximal number np_ℓ of polynomials p, r or G required to build a $z_\ell(x)$, $z_\ell'(x)$ or $\bar{Q}_\ell(x)$, respectively, is, as indicated by the eqs. (4.1) , (4.3) and (4.6):

$$(4.7) \quad np_\ell = -1 + \sum_{k=1}^{\ell} (k+1) = \frac{1}{2}\ell(\ell+3) - 1^{*)}$$

The maximal number of coefficients of these np_ℓ complete dense polynomials is:

$$(4.8) \begin{cases} nc_\ell = -\ell + \sum_{k=1}^{\ell} \sum_{m=1}^{k+1} (\ell-k+1) = \\ \quad = \frac{1}{6}\ell\{\ell(\ell+6) - 1\}. \end{cases}$$

The maximal number of polynomials u, also completely dense, requested for an $F_\ell(x)$, is $mp_\ell = np_\ell + 1$; the corresponding number of coefficients is $mc_\ell = nc_\ell + \ell$.

So we require a total number of TP_L polynomials and TC_L coefficients, where

$$(4.9.a) \begin{cases} TP_L = 2NP_L + MP_L + GP_L, \\ TC_L = 2NC_L + MC_L + GC_L \end{cases}$$

if

$$(4.10) \begin{cases} NP_L = \sum_{\ell=1}^{L} np_\ell = \frac{1}{6}L\{L(L+6) - 1\} = nc_L, \\ MP_L = NP_L + L \\ NC_L = \frac{1}{24}L(L+1)\{L(L+9) + 2\}, \\ MC_L = NC_L + \frac{1}{2}L(L+1). \end{cases}$$

Notice that (4.9.a) states that NP_L polynomials p and r and MP_L polynomials u are finally stored. In view of the structure of eq. (2.3) this is inevitable, at least for the p-polynomials. $GP_L = np_L$ and $GC_L = nc_L$, i.e. we only have to replace ℓ by L in eqs. (4.7) and (4.8) respectively, since the G-polynomials are only temporarily needed.
If the r- and u-polynomials are recomputed each time they are required for the construction of a G-polynomials we save a lot of storage, as reflected by

$$(4.9.b) \quad TP_L = NP_L + GP_L, \quad TC_L = NC_L + GC_L.$$

Table 2 shows some figures for different L-values.

L	NP_L	MP_L	GP_L	TP_L^a	TP_L^b	TC_L^a	TC_L^b
5	45	50	19	159	64	330	135
9	201	210	53	665	254	2091	816
10	265	275	64	869	329	2960	1145
15	785	800	134	2504	919	11765	4405
17	1105	1122	169	3501	1274	18241	6766
18	1293	1311	188	4085	1481	22326	8047
20	1730	1750	229	5439	1959	32495	11915
25	3225	3250	349	10049	3574	72775	26300

Table 2

The equations (4.7), ..., (4.10) describe upper-bounds. The experiments showed that certain polynomials always vanish. We found

$$(4.11) \begin{cases} p_{\ell,\ell,\ell+1} \equiv 0 \text{ and } p_{\ell,\ell,\ell} \equiv 0, \\ \qquad \ell = 1, \ldots, L, \text{ implying} \\ r_{\ell,\ell,\ell+1} \equiv 0, \; r_{\ell,\ell,\ell} \equiv 0, \\ u_{\ell,\ell,\ell+1} \equiv 0, \; u_{\ell,\ell,\ell} \equiv 0. \end{cases}$$

4.2.1. An ALTRAN-model

In ALTRAN [3,10] an algebraic variable has the form $\mu\nu^{-1}\prod_j \phi_j^{\varepsilon_j}$, where μ,ν are relatively prime monomials (ν positive), the ϕ_j are distinct multinomials and the ε_j are short nonzero integers.
A Gaussian rational c can be represented as a pair (real(c), imag (c)). A rational r = num/den $\in Q$ requires 2n words of storage if $|\text{num}| \le 10^{10n} - 1$, $|\text{den}| \le 10^{10n} - 1$ and $1 \le n \le 10$. Hence such a pair requires 4n words of storage. Alternatively, c can be represented as an algebraic in the indeterminate $i = \sqrt{-1}$, i.e. as $\mu\nu^{-1}(\alpha + i\beta)$. Again 4n words are required to store the integers μ, ν, α, β and 2 extra words for the exponents of i. It, however, might occur that $\mu = 1$, $\nu = 1$, $\alpha = 0$ or $\beta = 0$. So, in general, a reasonable estimation is

*) Since $p_{1,1,1} \equiv 0$ it follows (by induction on ℓ) that all $p_{\ell,1,1} \equiv 0$, $r_{\ell,1,1} \equiv 0$ and $G_{\ell,1,1} \equiv 0$. This is not necessarily true for $u_{\ell,\hat{1},1}$ of eq. (4.4).

$(3+\varepsilon)n+2$ words if $0 \le \varepsilon \le 1$.

A polynomial like $p_{\ell,k,m}$ will have the form $\mu\nu^{-1} \sum_{j=0}^{\ell-k} \alpha_j x^j + i\beta_j x^j$, which requires at most $2\{((\ell-k+1)n + (\ell-k+1)\}$ words, since again $\mu = 1$, $\nu = 1$, $\alpha_j = 0$ or $\beta_j = 0$ might be possible.

Let us now consider the set of all polynomials and coeefficients for some L. Performing all operations on polynomials, as we did, implies that almost A1 words are required, where

$$A1 = 2\{(TC_L + TP_L)n + TC_L\}.$$

Operating instead only with coefficients requires A21 words (when representing them as algebraics in i) or A22 words (when performing all operations on pairs of rationals), where

$$A21 = \{(3+\varepsilon)n+2\}\ (TC_L - GC_L),$$

$$A22 = 4n(TC_L - GC_L).$$

Notice that A1 is an upperbound, A21 gives minimal requirements when $\varepsilon = 0$ and A22 reflects a fixed amound of storage.

As stated before some of the polynomials vanish. This does not affect A22, as long as we have not a proof for this phenomenon. A22 can also be considered as a special case of A21 ($\varepsilon = (n-2)/n$). So when choosing between A1 and A21 we in fact ask for which par (L,n) holds, assuming $\varepsilon = 0$, that $A1 < A21$. Simplifying this inequality somewhat results in the same question for $TC_L - 2TP_L > 4GC_L = 4NP_L$ (i.e. $n \ge 2$, at least double precision integers).

Consultation of table 2 shows that for approach (a), no recomputations, the polynomial representation is best for $L > 9$. If we decide to recompute the r- and u-polynomials each time they are required, a polynomial representation preference is only justified if $L > 17$.
This question is of course related to the storage requirements. Let

$$S_n(L) = \left\lceil \frac{2n(TC_L + TP_L) + 2TC_L}{1000} \right\rceil$$

be the number of K (= 1000) words for an integer-length of n words. Table 3 gives some figures for this A1-model.

n \ L	10	15	20	25	
2	22	81	217	477	
4	37	138	369	809	
6	52	195	521	1140	(a)
8	68	252	672	1471	
10	83	309	824	1803	
2	9	32	80	173	
4	15	52	135	292	
6	20	73	191	412	(b)
8	26	94	246	499	
10	32	116	301	611	

Table 3: some $S_n(L)$-values

When running an ALTRAN-program the value n is fixed via a JCL-instruction. Once n is chosen all coefficients of algebraics have a fixed length of n words, when assigning the attribute LONG. This implies that the largest coefficient determines the storage requirements.
However $n \le 10$. Hence, as soon as the coefficient growth exceeds n words (this means for a PDP10, for instance, a coefficient length bound of 100 decimal digits) we are in trouble. And our experiments, reflected by the $cg(\ell)$ of section 4.1, made clear that we were in trouble indeed. Recomputation of intermediate results or another representation, thus saving storage (and swopping time) does not help in combatting the incooperative behaviour of the coefficients. If one insists on using ALTRAN one can try to simulate the commonly used list structure for long integers by introducing some univariate polynomial representation, where the indeterminate is in fact some power of 10. But, just to mention one aspect, this will increase the storage requirements enormously . A consultation of table 3 and of $cg(\ell)$ learns that such an approach is almost fruitless. Shortly after my first confrontation with this phenomenon I gave a talk, entitled "How discouraging is ALTRAN?" When Stan Brown read the abstract, in which was said that such a simulated list structure might be considered,,

he send me a kind warning. A citation: "... From our conversations I am aware of your efforts to produce rational approximations with exact rational number coefficients. When carried to high orders, such approximations inevitably lead to a combinatorial explosion of the coefficients, and in ALTRAN this causes coefficient overflow before the order can become very large. I don't blame you at all for being discouraged by this phenomenon, but I fear that your efforts to avoid it by representing integers as polynomials or truncated power-series will lead you into a quicksand of complexity and inefficiency ...". Of course, I neglected his advice and did some experiments in this direction, be it in a different, but, arithmetically spoken, similar context: resultants of polynomials [17]. I simply wanted to know how discomfortable basic facilities can be for this type of practical problems. Now I agree, but let me return to this question in section 6.

4.2.2. A REDUCE-model

A natural question is - before starting to try to improve the algorithm - if the difficulties are essentially system independent or not, i.e. is it possible to get a satisfactory solution using a system with arbitrary precision arithmetic, such as Reduce [11].
Hence let us make an estimation of the discomfort using the expected number of Gaussian rationals in combination with the expected growth of the coefficients and the provided information about BIGNUM's.

The ALTRAN-experiments showed that the length of the integers, occuring in the $H_{\ell}'(0)$ and $F_{\ell}(\infty)$, is roughly given by $\ell(\ell-3)$. These constants were obtained via summation, cf eq. (4.5). Let us therefore assume that during step ℓ of the iterative process the average length of the integers, occurring in the coefficients is given by $(\ell-1)(\ell-4)$. So a measure for the increase of the storage requirements, necessary for the p-polynomials generated during step ℓ, can be $m(\ell) = nc_{\ell}(\ell-1)(\ell-4)$, cf eq. (4.8). So, again, we accept a recomputation of the r- and u-polynomials, each time they are required during the construction of the G-polynomials. Although these polynomials are only temporarily required, they must be stored when executing step ℓ. A measure for these G-requirements is $GC_{\ell}(\ell-1)(\ell-4) = nc_{\ell}(\ell-1)(\ell-4) = m(\ell)$. If we neglect the storage requirements for all $\ell < \alpha$ a measure $T_m(L)$ for the total requirements for $\ell = \alpha, \ldots, L$ is thus given by:

$$T_m(L) = m(L) + \sum_{\ell=\alpha}^{L} m(\ell) \geq m(L) + M(L) - M(\alpha+1),$$

where

$$M(L) = \int_0^L m(\ell)\,d\ell.$$

Some straightforward computations show, assuming $\alpha = 5$, that

$$T_m(L) \geq C_m(L) = \frac{L^3}{360}\{((10L+72)L-345)L-1040\}.$$

The quantity nc_{ℓ} denotes the number of Gaussian rationals, introduced during step ℓ to construct new p-polynomials. So the number of integers is at least $3nc_{\ell}$. The BIGNUM-representation requires 2 words to represent a group of 10 decimal digits. So, if $T_n(L)$ denotes a measure for the number of storage words, accociated with $T_m(L)$, it is reasonable to assume

$$T_n(L) = \frac{3}{5}T_m(L) \geq \frac{3}{5}C_m(L).$$

Or using $T_k(L)$ to denote an estimation for the storage requirements, by taking K (= 1000 words) as a unit, gives:

$$T_k(L) \geq C_k(L) = \lceil 3\, C_m(L)/5000 \rceil.$$

Table 4 shows some $C_k(L)$-values.

L	$C_k(L)$	L	$C_k(L)$	L	$C_k(L)$
7	2	14	164	20	1345
8	6	15	247	21	1792
9	11	16	361	22	2355
10	22	17	517	23	3058
11	39	18	724	24	3926
12	65	19	995	25	4990
13	105				

Table 4

How realistic are these figures?
Table 3 shows, when using alternative (b), that $s_6(10) = 20 < C_k(10) < 26 = s_8(10)$. In view of our empirical results, see table 1, the quantities $s_6(10)$ and $s_8(10)$ are quite realiable. ALTRAN's datastructures are based on format tables, thus avoiding the neccessity of using lots of pointers to represent long integers. A disadvantage is the waste of storage when the integer length is quickly increasing. But, apparently, this waste is nearly compensated by the avoidance of pointers, as required in Reduce. An objection can be that an unfair comparison is made between a coefficient-model (Reduce) and a polynomial-model (ALTRAN). But we simply assume that the extra storage, required to represent complete polynomials is compensated by the eventually not optimal presentation of the coefficients. Even if we relaxe the predicted growth of the integers, by taking as average length, $\alpha(\ell - 1)(\ell - 4)$, where α is a constant such that $0.5 \le \alpha \le 1$, resulting in $\alpha C_k(L)$ instead of $C_k(L)$, this does not essentially affect the space complexity of the algorithm, being $O(L^6)$. So we must conclude that in terms of core requirements this improved algorithm is still an insuperable handicap.

5. A MUCH BETTER METHOD

An alternative might be to analyze the asymptotic behaviour of the solution at infinity. Dijkstra constructed asymptotic series which are uniformly convergent over the entire range $[0,\infty]$, so that a rigorous, simple existence proof can be given near rigid rotation [5].
An important aspect of this approach is the far less complicated structure of the solution, i.e. its predictable superior computational behaviour over the previously mentioned methods.
We give only an outline of this approach, just to provide enough information to give a convincing indication of the storage requirements.

So let us return to eq. (2.2):

$$(5.1) \quad H'' + 2FH' = H^2 + 1, \; H(\infty) = i$$

where

$$2F(x) = \int_0^x \{H(t) + \overline{H}(t)\}dt$$

and

$$H(x) = F'(x) + iG(x).$$

For $x = 0$ the boundary conditions are $H = i\sigma$, $F = 0$. The Rogers and Lance linearization of section 4.1 was obtained by the substitution $\sigma = 1 + \delta$.
Let us define $F(\infty) = A$, where A is some real constant. This suffices to agree that eq. (5.1) can be rewritten as

$$(5.2) \quad H_1'' + 2AH_1' - 2iH_1 = 0$$

if the original equation is linearized at infinity, i.e. if we take $F = A + F_1$, $H = i + H_1$ and if we neglect squares of the perturbations. Via the characteristic equation, associated with eq. (5.2)

$$(5.3) \quad \mu^2 + 2A\mu - 2i = 0$$

we obtain a second order approximation

$$(5.4) \quad H(x) = i + C_1 e^{\mu x} + C_2 e^{\mu_2 x}$$

where

$$(5.5) \quad \mu = P + i\,\Omega = -A - (A^2+2i)^{1/2}, \; \mu_2 = 2P,$$

C_1 is a (free) complex constant and

$$(5.6) \quad C_2 = -\mu C_1\overline{C}_1(\mu_2^2 + 2A\mu_2 - 2i)^{-1} / \overline{\mu}.$$

Before we continue, we clarify the relation to the RL-approach. At rigid rotation $(\delta = 0)$ the solution is $H \equiv i$. Hence the constants A and C_1 ought to vanish (and thus also C_2). However if $\delta \neq 0$ is small the solution differs from $H = i$ by an amount of $O(\delta)$, implying that both A and C_1 are $O(\delta)$ and C_2 is $O(\delta^2)$. Observe that, in terms of A and C_1, the boundary values at the disk are still free. Although eq. (5.4) reflects an asymptotic approximation valid voor $x \to \infty$ it is possible to reach the disk if δ is small and if higher order terms are neglected:
According to eq. (5.4) a first order approximation at the disk is $H(0) = i + C_1 + O(C_2)$. Since $H(0) = i + i\delta$ we find $C_1 = i\delta + O(C_2)$. But $C_2 = O(C_1^2)$. So we can take $C_1 = i\delta + O(\delta^2)$.
Then a consultation of eq. (5.5) shows that $\mu = P + iQ = -1 - i + O(A) = -1 - i + O(\delta)$. Substitution of these results gives:

$$(5.7) \quad H(x) = i + i\delta e^{-(1+i)x} + O(\delta^2).$$

But eq. (5.7) can be read as $H(x) = i + \delta H_1$, where $H_1 = p_{1,1,2}\,\xi_{1,2}$ (according to definition 4.1 and theorem 4.1).

Continuation of this approach, now putting $C_1 = i\delta$, $\mu = -(1+i)$, $\mu_2 = -2$, $A = 0$ results in

$$(5.8)\quad C_2 = \frac{1-2i}{10}\,\delta^2.$$

Again taking $H(0) = i(1+\delta) = i + C_1 + C_2$ now gives

$$(5.9)\quad C_1 = i\delta - \frac{1-2i}{10}\,\delta^2$$

with

$$(5.10)\quad A = \frac{1}{2}\,\delta + O(\delta^2)$$

We find

$$(5.11)\quad \mu = -(1 + \frac{1}{2}\,\delta + i) + O(\delta^2)$$

and

$$(5.12)\quad \mu_2 = -2 + O(\delta)$$

So

$$(5.13)\quad H(x) = i + (i\delta - \frac{1-2i}{10})e^{-(1+\frac{1}{2}\delta+i)x} + \frac{1-2i}{10}\,\delta^2 e^{-2x}.$$

Replacing in eq. (5.13) $e^{-\frac{1}{2}\delta x}$ by its Taylor-series results in

$$(5.14)\quad H(x) = i + i\delta e^{-(1+i)x} + \frac{\delta^2}{10}\,\{(1-2i)e^{-2x} + (2i-1-5x)e^{-(1+i)x}\} + O(\delta^3).$$

Rewriting eq. (5.14) as $H(x) = i + \delta H_1 + \delta^2 H_2$ allows to formulate eq. (5.14) in terms of definition 4.1 and theorem 4.1 as

$$H(x) = i + \delta(p_{1,1,2}\xi_{1,2}) + \delta^2(p_{2,1,2}\xi_{1,2} + p_{2,2,1}\xi_{2,1}) + O(\delta^3)$$

which is again the RL-formulation, cf eq. (4.2) of section 4.1.

Remark: Obviously eq. (5.13) is simpler than eq. (5.14). The essential difference is apparantly caused by the Taylor expansions of certain exponential functions. The RL-formulation forces to refine iteratively complete Taylor expansions, the Dijkstra-approach demands only for the repeated refinement of constants (C's) and arguments of exponentials (μ's). These quantities (C and μ) are independent of the variable x, implying that required differentiations of H(x) or integrations of H(x) and $\overline{H}(x)$ are straightforward.

How to continue this process, i.e. how to give a general description, allowing to define an algorithm? Let

$$(5.15)\quad H = i + \textstyle\sum_\alpha C_\alpha e^{\mu_\alpha x}$$

where α is some (double)summation index. Then

$$(5.16)\quad 2F(x) = 2A + \textstyle\sum_\alpha \{\frac{C_\alpha}{\mu_\alpha}\, e^{\mu_\alpha x} + \frac{\overline{C}_\alpha}{\overline{\mu}_\alpha}\, e^{\overline{\mu}_\alpha x}\}.$$

And thus, using eq. (5.1):

$$(5.17)\quad \textstyle\sum_\alpha (\mu_\alpha^2 + 2A\mu_\alpha - 2i)C_\alpha e^{\mu_\alpha x} = \sum_\beta\sum_\gamma (1 - \frac{\mu_\beta}{\mu_\gamma})C_\beta C_\gamma e^{(\mu_\beta+\mu_\gamma)x} - \sum_\beta\sum_\gamma C_\beta \overline{C}_\gamma (\frac{\mu_\beta}{\overline{\mu}_\gamma}) e^{(\mu_\beta+\overline{\mu}_\gamma)x}.$$

If eq. (5.17) is satisfied then eq. (5.15) is a (formal) solution, satisfying the boundary conditions at infinity if $Re(\mu_\alpha) < 0$.

Let us neglect the righthand side of eq. (5.17) to obtain the first value of μ_α (it is convenient to denote α by (1.0)). Hence:

$$(5.18)\quad \{\mu^2(1,0) + 2A\mu(1,0) - 2i\}C(1,0) = 0.$$

If we use eqs.(5.3) and (5.5), i,e. if we take $\mu(1,0) = \mu = P + i\Omega$, then C(1,0) is apparently a free constant, previously denoted by C_1.

Now we consider the second order terms by putting $\beta = \gamma = (1,0)$, resulting in two possible exponents $\mu(1,0) + \mu(1,0)$ and $\mu(1,0) + \overline{\mu}(1,0) = 2P$. The first combination delivers a vanishing coefficient $\{1 - \mu(1,0) / \mu(1,0)\}$. If we denote the latter by $\mu(2,1) = 2P$ we find:

$$(5.19)\quad C(2,1) = -C(1,0)\overline{C(1,0)}\,\frac{\mu}{\overline{\mu}}\,\{\mu^2(2,1) + 2A\mu(2,1) - 2i\}^{-1}.$$

Eq. (5.19) is again in agreement with eg. (5.6) if we replace C(2,1), C(1,0) and $\mu(2,1)$ by C_2, C_1 and μ_2 respectively.
The third order terms allow the combinations $\mu(3,1) = \mu(2,1) + \mu(1,0) = 3P + iQ$ and $\mu(3,2) = \mu(2,1) + \bar{\mu}(1,0) = 3P - iQ$. Dijkstra showed that in general holds:

(5.20) $\mu(n,m) = nP + (n-2m)iQ$,
where n = 1,2.... and if n = 1 then
m = 0 else m = 1,..., n-1.

Remark: Eq (5.20) indicates a refinement of definition 4.1 and theorem 4.1 and confirms our empirical results, as given by eq (4.11). Observe the striking similarity between the $\xi_{k,m}$ of definition 4.1. and the $e^{\mu(k,m)x}$ of eqs. (5.15) and (5.20) where $\mathrm{Re}(\mu(k,m)) < 0$ must hold.

Now knowing the μ_α, at least in principle since the P and Q depend on A, eq. (5.17) indicates how to obtain the C_α:
According to eq. (5.18) C(1,0) is still a free constant. Setting $\alpha = (n,m)$, $n > 1$ and $1 \le m \le n-1$, and consulting eq. (5.17) learns that

$$(\mu_\alpha^2 + 2A\mu_\alpha - 2i)C_\alpha = \sum_\beta\sum_\gamma(1 - \frac{\mu_\beta}{\mu_\gamma})C_\beta C_\gamma - \sum_\beta\sum_\gamma \frac{\mu_\beta}{\bar{\mu}_\gamma} C_\beta \bar{C}_\gamma$$

with $\beta = (n_1,m_1)$ and $\gamma = (n_2,m_2)$ such that $\mu_\alpha = \mu_\beta + \mu_\gamma$ and $\mu_\alpha = \mu_\beta + \bar{\mu}_\gamma$ allows to compute C(n,m) from the previously computed C's.

Finally we remark that the value of the free constants A and C(1,0) depends on the boundary conditions at the disk (x = 0), where ought to hold $H(0) = i(1+\delta)$ and $F(0) = 0$. Hence

(5.21) $\sum_\alpha C_\alpha = i\delta$ and $\sum_\alpha \mathrm{Re}(C_\alpha/\mu_\alpha) = -A$

Summarizing, we have to solve the following infinite implicit system of non-linear equations:

$$(5.22)\quad \begin{cases} \mu = P + iQ = -A-(A^2+2i)^{\frac{1}{2}} = \mu(1,0) \\ \mu_\alpha = nP + (n-2m)iQ,\ \alpha = (n,m) \\ n = 1,2,3,\ldots;\ m = \min(1,n-1),\ldots,n-1 \\ (\mu_\alpha^2+2A\mu_\alpha-2i)C_\alpha=\sum_\beta\sum_\gamma(1-\frac{\mu_\beta}{\mu_\gamma})C_\beta C_\gamma - \\ \qquad \sum_\beta\sum_\gamma \frac{\mu_\beta}{\bar{\mu}_\gamma} C_\beta\bar{C}_\gamma \\ \sum_\alpha C_\alpha = i\delta\ ;\ \sum_\alpha \mathrm{Re}(C_\alpha/\mu_\alpha) = -A \end{cases}$$

Assuming this is possible it is quite obvious how to obtain H'(0) and F(∞). But is it possible? For general values of δ this can be difficult. In addition it must be recalled that eq. (5.15) reflects an expansion of the solution for large values of x, i.e. we cannot expect it to hold at the disk for general values of δ. So implementing the boundary conditions (5.21) is questionable for general δ. But near rigid rotation ($|\delta| \ll 1$), our main interest, these arguments do not hold, since a rational truncation method can be developed for system (5.22), provided δ is small.
The truncation is based on the C-behaviour. Since $C(n,m) = O(C_1^n)$ and if $C_1 = O(\delta)$ then $C(n,m)=O(\delta^n)$ we have not to take into account the C(n,m) with $n > N$ if we want to compute the relevant quantities up to order δ^N (and as we actually did before).

To illustrate this we produce the third order approximation, using eq. (5.22). We start improving A (given as $A = \frac{1}{2}\delta + O(\delta^2)$ by eq. (5.10)) using

$$A = -\sum_\alpha \mathrm{Re}(C_\alpha/\mu_\alpha) = -\mathrm{Re}\{\frac{C(1,0)}{\mu(1,0)} + \frac{C(2,1)}{\mu(2,1)}\}+O(\delta^3)$$

From eqs. (5.8), (5.9), (5.11) and (5.12) we derive

$$A = \frac{1}{2}\delta - \frac{3}{20}\delta^2 + O(\delta^3).$$

Once having this new value of A we can refine the μ_α's, $\alpha = (n,m)$, $n \le 3$, using system (5.22) and resulting in:

$$\mu(1,0) = -1-i - \frac{1}{2}\delta + \frac{7+5i}{80}\delta^2 + O(\delta^3)$$
$$\mu(2,1) = -2 - \delta + O(\delta^2)$$
$$\mu(3,1) = -3 - i + O(\delta),\ \mu(3,2) = -3 + i + O(\delta);$$

Then, again using (5.22) we compute C(2,1), C(3,1) and C(3,2) followed by an application of $\sum_\alpha C_\alpha = i\delta$ to obtain a refined value for

$C(1,0) = i\delta - \{C(2,1) + C(3,1) + C(3,2)\} + O(\delta^4)$.

Finally we get improved values for

$H'(0) = \sum_\alpha C_\alpha \mu_\alpha + O(\delta^4)$, $F(\infty) = A = -\sum_\alpha \mathrm{Re}(C_\alpha/\mu_\alpha)$

with $\alpha = (n,m)$, $n \leq 3$.

These values, in accordance with the results of Rogers and Lance (see table 1), are

$$H'(0) = (1-i)\delta + \frac{1-2i}{10}\delta^2 + \frac{5+9i}{400}\delta^3 + O(\delta^4)$$

$$F(\infty) = A = \frac{1}{2}\delta - \frac{3}{20}\delta^2 + \frac{7}{160}\delta^3 + O(\delta^4)$$

Without going into details, an analysis, similar to that of section 4.2., will show that the space compexity is now $O(L^4)$. This implies that approximately 285 ($\approx 5000^{2/3}$) k is required when using Reduce to produce the first 25 coefficients. However, Gagert had just finished a SIMULA-package*) for multiple precision floating point arithmetic, using the class-concept of this language. It is similar to packages made by Brent [2] or Sasaki [14]. The exact information obtained via ALTRAN provided enough information to get insight in "the floating point-behaviour" of the algorithm.

This time we obtained up to 18 terms. However,due to intensive swopping activities, a lot of CPU-time was used. Later on Gragert wrote another package, also in SIMULA *), to operate on truncated power series with Gaussian floating point coefficients. This package finally delivered in only 15 minuted CPU-time the first 30 constants $H'_\ell(0)$, $F_\ell(\infty)$ in at least 10 reliable decimal digits [6].

6. CONCLUSIONS

Some of Jenks' statements in the section SIGSAM and SIAM of his PUSH 80 mainly motivated me to write this contribution. A citation "....MACSYMA, however, provides an interesting solution to this interface problem. It translates FORTRAN code, much of the large numerical analysis library IMSL, in fact, into MACLISP (yes, the resulting code is efficient). And since LISP has a resident compiler, dynamically created symbolic expressions can be subsequently swept through the compiler. There ends the problem...". I disagree, since the character of numerical problems is too diverse to justify such a uniform approach. Although an attempt to classify these problems and possible associated strategies is dangerous too and certainly violates reality, it might be worthwhile trying to give some indications.

There is no doubt that a routinely use of computer algebra in analyzing numerical problems and in preparing FORTRAN code is often quite profitable. But in a sense we are prisoners of the existing facilities. Either we are doomed to operate with exact number systems or we are condamned to live in a floating point vegetation.

Exact approaches do not always cause problems, since constants not necessarily have to explode during the computations. And when it happens during some intermediate stages of the computation we often know that the constants occuring in the final results are bounded,thus allowing to improve the algorithms to govern these unwanted problems. But there is a category of problems which is essentially different. Explosion of constants or alternatively (rigorous)error propagation belong to it. For instance: Perturbation problems in combination with boundary conditions.

Another citation from Brown's earlier mentioned letter: "....The datastructure (of ALTRAN) was inherited from Alpak, which was coded in assembly language, and was chosen for reasons of efficiency. Using an array of fixed length integers for the coefficient blocks allows all the operations to run at lightning speed, provided that the maximum coefficient is not very large. For the case where some coefficients are small and others are quite large, I always intended to have an alternative representation using a separate block for each coefficient , and a block of pointers to them, but we never felt a compelling need to implement this concept.....".

He is right, I think. The bounded length of integers in ALTRAN can be considered as one of the charmes of the system. When dealing with problems with an inherent integer explosion ALTRAN behaves like a microscope: one is quickly running into trouble, thus stressing the malbehaviour of the algorithm.

*) written documentation is not yet available.

If John Fitch guesses fluid mechanics to be the next field to take to computer algebra [9], he apparently neglects the lack of "user-friendliness of algebra systems" for this user-category. Intuition, or perhaps the wish to force reality, plays an important role in the interpretation of numerical results in fluid mechanics. However, I hope I have been illustrating that intentions to do it more objective can be time consuming and sometimes even discouraging. Is the price too high, if one classifies such attempts as software development costs? Although Dijkstra's final approach is elegant, one can wonder if such a time-consuming effort had been necessary if algebra systems were offering facilities to carry out reliable instead of exact arithmetic. A consultation of table 2 shows, that, when programming alternative (b), only 265 K of words would have been necessary to produce the first 25 coefficients if an average of 10 words had been sufficient to represent the Gaussian rationals in a reliable way. I have not yet a clear cut solution, but indications in literature exist that research in this direction is certainly profitable.
To conclude with, an illustrative example of this might be the progress made in the calculation of zeros of unvariate polynomials.
Verbaeten [18], for instance, when analyzing the SAC-1 subsystem for computing the real zeros of univariate polynomials with integer coefficients, discovered that over 90% of the total time was spent in interval refinement. Part of his improvements were based on approximation of rational numbers to govern the growth of numberlengths. Collins too, did similar and other experiments. [4].

REFERENCES

[1] R.J. Bodonyi, On rotationally symmetric flow above an infinite rotating disk, Journal Fluid Mechanics 67, (1975), p. 657.

[2] R.P. Brent, A FORTRAN multiple-precision arithmetic package, ACM TOMS 4 (1978), p. 57.

[3] W.S. Brown, ALTRAN user's manual, Bell Laboratories, Murray Hill (1977).

[4] G.E. Collins, Infallible Calculations of polynomial zeros to specified precision, Mathematical Software III (J.R. Rice, editor), Academic Press, (1977), p. 35.

[5] D. Dijkstra, Some Contributions to the solution of the rotating disk problem, Memorandum No. 205, Department of Applied Mathematics, Twente University of Technology (1978).

[6] D. Dijkstra and P.K.H. Gragert, Production and application of Taylor series for the rotating disk problem, in preparation.

[7] D. Dijkstra and P.J. Zandbergen, Some further investigations on non-unique solutions of the Navier-Stokes equations for the Karman swirling flow, Archives Mechanics 30 (1978),p.411.

[8] D.J. Evans, The rotationally symmetric flow of a viscous fluid in the presence of an infinite rotating disk with uniform suction, Q.J.M.A.M. 22 (1969), p. 467.

[9] J. Fitch, The application of symbolic algebra to physics - a case of creeping flow, Symbolic and Algebraic Computation (E.W. Ng, editor), Springer LNCS series No. 72 (1979), p. 30.

[10] A.D. Hall, The ALTRAN system for rational function manipulation, - A survey, Comm. ACM 14 (1971), p. 517.

[11] A.C. Hearn, Reduce 2 user's manual, 2nd edition, University of Utah, Report No. UCP-19 (1973).

[12] D. Jenks, PUSH 80, ACM SIGSAM Bulletin Vol.13 No. 4 (1979), p. 3.

[13] M.H. Rogers and G.N. Lance, The rotationally symmetric flow of a viscous fluid in the presence of an infinite rotating disk, Journal Fluid Mechanics 7 (1960), p. 617.

[14] T. Sasaki, An arbitrary precision real arithmetic package in Reduce, Symbolic and Algebraic computation (E.W. Ng , editor),Springer LNCS series No. 72 (1979), p. 358.

[15] H. Schlichting, Boundary Layer Theory, McGraw Hill (1968).

[16] J.A. van Hulzen, Production of exact Taylor coefficients for the rotating disk problem using an algebra aystem, Memorandum No. 183, Department of Applied Mathematics, Twente University of Technology (1977).

[17] J.A. van Hulzen, A note on solving systems of polynomial equations with floating-point coefficients, Symbolic and Algebraic Computation (E.W. Ng, editor), Springer LNCS series No. 72 (1979), p. 346.

[18] P. Verbaeten, Computing Real zeros of polynomials with SAC-1, ACM SIGSAM Bulletin vol. 9 No. 2 (1975) p. 8.

[19] P.J. Zandbergen and D. Dijkstra, Non-unique solutions of the Navier-Stokes equations for the Karman swirling flow, J. Engg. Math. 11 (1977), p. 167.