XXVII. COMPUTATION RESEARCH*

Research Staff

Martha M. Pennell
Elaine S. Brown
Heather S. Davis
Joan Harwitt
Elaine C. Isaacs
Eleanor C. River

A. POLYNOMIAL ROOT FINDING

Work has been progressing on the MAD and FORTRAN IV version of ROOTS, a polynomial-with-complex-coefficients-root-finding subroutine.[1] Our MAD program is essentially completed and is being used as a prototype for the FORTRAN IV subroutine, which is now being debugged on the M.I.T. Computation Center's 360/65.

In the cases that we have tried, we have solved the problem of round-off[1] by testing

$$\frac{|y|}{|x|} < 10^{-4} \tag{1}$$

$$\frac{|x|}{|y|} < 10^{-4} \tag{2}$$

where $z = x + iy$ is the final iterate to which the program has converged as a root of the polynomial $f(z)$. This testing is done outside the iteration loop, but before the coefficients of the reduced polynomial are calculated. If neither (1) nor (2) is true, $x + iy$ is assumed to be indeed a root, and synthetic division continues. If, however, (1) is true, then $|f(z=x+iy)|$ is compared with $|f(z=x)|$. If the latter is smaller than the former, we consider y to be round-off error and synthetic division continues with $z = x$ as the true root. Similarly, if (2) holds, we consider x to be round-off and $z = y$ to be the true root. This method resulted in the following roots to the polynomial $3x^5 - x^4 + 6x^3 - 2x^2 + 3x - 1 = 0$:

```
.3333334  +0          i
   0      +.9995337   i
   0      -1.00000    i
   0      -.9999999   i
   0      +1.0000466  i
```

whose true roots are $\frac{1}{3}$, $\pm i$ (double root).[2]

This same technique is used in POLRT, the SSP[3] subroutine which computes the real and complex roots of a real polynomial.

---

*This work was supported in part by the Joint Services Electronics Programs (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DA 28-043-AMC-02536(E).

The program uses the Newton-Raphson technique, which involves no root taking and consequently must always start with an initial complex guess to find a complex answer. POLRT fails after no convergence with 500 iterations on the 5 starting values:

-.1000101	-.0500101
.500101	1.000101
-10.00101	-5.00101
50.00101	100.0101
-1000.101	-500.0101.

We found that POLRT failed on the following polynomial

$$3x^{35} - 4x^5 + 1 = 0,$$

30 of whose roots are known to lie within the unit circle, and none are double roots. POLRT succeeded, however, when the polynomial was divided through by the obvious root $x = 1$, and also succeeeded when the substitution

$$y = x^5$$

resulted in the following polynomial:

$$3y^7 - 4y + 1 = 0.$$

We are planning to study the relative merits of both methods.

Martha M. Pennell, Joan Harwitt

References

1. Martha M. Pennell and Joan Harwitt, "Roots, a Root-finding Subroutine Using Muller's Method," Quarterly Progress Report No. 84, Research Laboratory of Electronics, M.I.T., January 15, 1967, pp. 352-354.
2. Ibid., see Eq. 1, p. 353.
3. "System/360 Scientific Subroutine Package (360A-CM-03X) H20-0166-1," IBM Applications Program, n. d., pp. 84-85.

B. MODIFICATION OF THE IBM SCIENTIFIC PACKAGE FOR GENERAL USE

A basic problem in the IBM Scientific Package for os/360 is dimensioning of two-dimensional arrays. The manual states that in the main program "If the matrix is smaller than the dimensioned area, the two forms of storage are not compatible."[1] For example, if a matrix is dimensioned A(50,50) and the amount to be used is only A(25,5) or A(25,25) the

Example 1

Original

Main Program

```
Dimension A(25,5)
  :
CALL PLOT (NO,A,N,M,NL,NS)[2]
```

Subroutine

```
Subroutine Plot (NO,A,N,M,NL,NS)
Dimension A(1)
  :
```

After Change

Main Program

```
Dimension B(50,50)
  :
K = 50
CALL PLOT (NO,B,N,M,NL,NS,K)
```

Subroutine

```
Subroutine Plot (NO,B,N,M,NL,NS,K)
Dimension B(K,K), A(5000)
I = 1
DO 39  J = 1,M
DO 39  K = 1,N
A(I) = B(K,J)
I = I + 1
39 Continue
  :
```

Example 2

Original

Main Program

```
Dimension A(25,5)
  :
CALL SIMQ(A, B, N, KS)[3]
```

Subroutine

```
Subroutine Simq(A, B, N, KS)
Dimension A(1), B(1)
  :
```

After Change

Main Program

```
Dimension A(50,50)
  :
M = 50
CALL SIMQ(A, B, N, KS, M)
```

Subroutine

```
Subroutine Simq(A, B, N, KS, M)
Dimension A(M, M) B(1)
  :
```

routines will not work. An obvious approach is to dimension the matrix exactly, but in some cases in which the matrix size varies within the program, this is not feasible.

Two methods have been used to correct this problem. In both cases the subroutine has been altered and one argument added to the calling sequence. This extra argument, tacked on at the end of the call, is the number of rows and columns in the dimensional matrix; it would be 50 in the example above. The restriction that the matrix be square is used to decrease the complexity of reprogramming. The dimensions for the portion of the matrix actually used are already included in the calling statement as arguments.

In the easier method, Example 1, the name of the matrix in the main program is changed from A to B. The change must also be carried through in the subroutine. Since the two-dimensional matrix B(50, 50) is now being brought through the calling sequence, it must be dimensioned as such in the subroutine. A no longer comes through the calling sequence, so it must be dimensioned as large as it could possibly be in the subroutine. Once this is done, a loop is used to place the portion of the B matrix that is to be used in the A vector. Since the program was originally written to use the vector A and all pertinent information is now in A, the subroutine may safely be used from this point on as it was.

In the second method, Example 2, the name of the matrix in the main program is not changed, but the dimension statement in the subroutine must be changed, since A is now brought through as a two-dimensional matrix. Once this is done, the indexing of A must be changed throughout the subroutine, and A treated as a two-dimensional matrix rather than a vector. This is a tricky procedure in most subroutines and there is less chance of error if the first method is used.

Our group is using both methods to alter subroutines from the IBM Scientific Package and has modified decks for SIMQ and PLOT available for general distribution.

Joan Harwitt

References

1. "System/360 Scientific Subroutine Package (360A-CM-03X), Version II," IBM Programmer's Manual, n. d., p. 4.
2. Ibid., p. 298
3. Ibid., p. 178.

C. CONFLUENT HYPERGEOMETRIC FUNCTION

Recently, we were presented with a request to evaluate the confluent hypergeometric function which has numerous applications in various branches of physics.

We were surprised to discover the paucity of published material on this topic. One of the more complete tables[1] of the confluent hypergeometric function, published in 1948,

was done by the Joint Computing Group of the Research Laboratory of Electronics.

Miss Elizabeth Campbell informs us that the evaluation of the confluent hypergeometric function had been, in 1948, a fairly large project. At that time, all calculations were done by hand; all figures were carried on a 10-bank desk calculator.

Several people were involved in the work of evaluating this function. Indeed, they spent a great deal of time finding ways to reduce the number of calculations needed for each set of values. Miss Campbell further estimated that to do all of the work would have taken one person an entire summer.

Today, the entire problem forms a rather short job. Using the series representation for the function

$$M(\alpha;\gamma;z) = 1 + \frac{\alpha}{\gamma}z + \frac{\alpha(\alpha+1)}{\gamma(\gamma+1)}\frac{z^2}{2} + \ldots = \sum_{n=0}^{\infty} \frac{\Gamma(\gamma)\,\Gamma(\alpha+n)\,z^n}{\Gamma(\alpha)\,\Gamma(\gamma+n)\,n!},$$

we found that the Fortran IV program to evaluate the confluent hypergeometric function took one person approximately 10 hours to write and debug.

The actual computer time needed to run the program was negligible. We used 191 values of α, γ, and z to test the accuracy of the program. Convergence occurred in a minimum of 3 steps for small values of α and z, γ being held constant, and in a maximum of 21 steps for larger values of α and z, γ again being held constant.

The program was run at the Computation Center, M.I.T., on the IBM 360/65. Compilation and run time for the program was 1min and 14.26 sec.

Admittedly, the existence of the earlier evaluation of the function simplified our check of the program. We plan to investigate further, however, the relative accuracy of man versus computer.

Elaine C. Isaacs

References

1. A. D. MacDonald, "Properties of the Confluent Hypergeometric Function," Technical Report 84, Research Laboratory of Electronics, M.I.T., November 18, 1948.

D. REPLACEMENT OF INFINITE LIMITS OF INTEGRATION BY FINITE ONES – AN ILLUSTRATION

We wish to integrate the function

$$E = \int_0^{\infty} re^{-r^2}\left[\int_0^{\infty} e^{-\lambda y^2} I_0(2r\sqrt{\lambda y})\,\frac{e^{\left[-(\ln y)^2/2\sigma_x^2\right]}}{y\sigma_x}\,dy\right]^{\rho/(1+\rho)} dr$$

$$= \int_0^\infty re^{-r^2} \left[\int_0^\infty L(r,y)\, dy \right]^{\rho/(1+\rho)} dr$$

for a given σ_x, λ, and ρ, where I_o is an I-Bessel function, and σ_x, λ, $\rho > 0$.

We do the integration numerically by evaluation of

$$A = \int_0^N re^{-r^2} \left[\int_0^{M(r)} L(r,y)\, dy \right]^{\rho/(1+\rho)} dr,$$

where N and M(r) are chosen to make $|E-A| < \epsilon$ for an arbitrary ϵ.

We shall show that we can find N and M(r) such that

$$P = \left| \int_N^\infty re^{-r^2} \left[\int_0^\infty L(r,y)\, dy \right]^{\rho/(1+\rho)} dr \right| < \frac{\epsilon}{2}$$

and

$$R = \left| \int_0^\infty re^{-r^2} \left[\int_{M(r)}^\infty L(r,y)\, dy \right]^{\rho/(1+\rho)} dr \right| < \frac{\epsilon}{2}.$$

It then follows that $|E-A| < \epsilon$. (Note that the functions re^{-r^2} and $L(r,y)$ are always positive when r and y are positive, so that absolute value signs are unnecessary.)

We first approximate $L(r,y)$ as follows. By definition,

$$I_o(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{2^{2n}(n!)^2}.$$

By Stirling's formula,

$$(n!) \approx e^{-n} n^n \sqrt{2\pi n}$$

$$2^{2n}(n!)^2 \approx e^{-2n} (2n)^{2n} \cdot 2\pi n$$

$$(2n)! \approx e^{-2n} (2n)^{2n} \sqrt{4\pi n}.$$

Therefore $(2n)! \leq 2^{2n}(n!)^2$ and

$$I_o(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{2^{2n}(n!)^2} \leq \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!} < \sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x.$$

Consequently, $I_o(2r\sqrt{\lambda}\,y) < e^{(2r\sqrt{\lambda}\,y)}$.

To estimate $B(y) = \dfrac{e^{\left[-(\ln y)^2/2\sigma_x^2\right]}}{y\sigma_x}$, we note that it is ≥ 0 on the interval $[0,\infty)$ and approaches zero as $y \to 0$ and as $y \to \infty$; It must therefore have a maximum on $[0,\infty)$.

$$\frac{dB(y)}{dy} = \frac{e^{\left[-(\ln y)^2/2\sigma_x^2\right]}}{y^2\sigma_x^2} \cdot \left(\frac{-\ln y}{\sigma_x} - \sigma_x\right).$$

So $\dfrac{dB(y)}{dy} = 0$ when $y = e^{-\sigma_x^2}$.

Therefore, letting

$$MAX = B\left(e^{-\sigma_x^2}\right) = \frac{e^{\sigma_x^2/2}}{\sigma_x},$$

we have $B(y) \leq MAX$ on $[0,\infty)$.

We now have

$$L(r,y) < e^{-\lambda y^2} \cdot e^{(2r\sqrt{\lambda}\,y)} \cdot MAX = MAX \cdot e^{-(\sqrt{\lambda}\,y - r)^2} \cdot e^{r^2}.$$

Therefore

$$P = \int_N^\infty re^{-r^2}\left[\int_0^\infty L(r,y)\,dy\right]^{\rho/(1+\rho)} dr$$

$$< (MAX)^{\rho/(1+\rho)} \int_N^\infty re^{-r^2}\left[\int_0^\infty e^{-(\sqrt{\lambda}\,y-r)^2} e^{r^2}\,dy\right]^{\rho/(1+\rho)} dr = Q.$$

Letting $z = \sqrt{\lambda}\,y - r$, we get

$$dz = \sqrt{\lambda}\,dy;$$

$$Q = (MAX)^{\rho/(1+\rho)} \int_N^\infty re^{-r^2}\left[\frac{e^{r^2}}{\sqrt{\lambda}}\int_{-r}^\infty e^{-z^2}\,dy\right]^{\rho/(1+\rho)} dr.$$

Since $\int_{-\infty}^\infty e^{-z^2}\,dy = \sqrt{\pi}$,

$$Q \leq \left(\text{MAX}\,\frac{\sqrt{\pi}}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} \int_N^\infty re^{(-r^2/[1+\rho])}\,dr$$

$$= \frac{(1+\rho)}{2}\left(\text{MAX}\frac{\sqrt{\pi}}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} e^{-N^2/(1+\rho)}.$$

It is clear then that for a given $\epsilon > 0$, we can choose N so large that

$$P < \frac{(1+\rho)}{2}\left(\text{MAX}\frac{\sqrt{\pi}}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} e^{-N^2/(1+\rho)} < \frac{\epsilon}{2}.$$

We have defined R as follows:

$$R = \int_0^\infty re^{-r^2}\left[\int_{M(r)}^\infty L(r,y)\,dy\right]^{\rho/(1+\rho)} dr.$$

Hence,

$$R < \left(\frac{\text{MAX}}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} \int_0^\infty re^{-r^2}\left[e^{r^2}\int_{(\sqrt{\lambda}\,M(r)-r)}^\infty e^{-z^2}\,dz\right]^{\rho/(1+\rho)} dr = S.$$

Now we can choose a fixed $M_o > 1$ such that

$$S_{M_o} = \left(\frac{\text{MAX}}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} \int_0^\infty re^{-r^2}\left[e^{r^2}\int_{M_o}^\infty e^{-z^2}\,dz\right]^{\rho/(1+\rho)} dr < \frac{\epsilon}{2}.$$

This follows, since for $M_o > 1$,

$$\int_{M_o}^\infty e^{-z^2}\,dz < \frac{1}{M_o}\int_{M_o}^\infty ze^{-z^2}\,dz = \frac{e^{-M_o^2}}{2M_o}.$$

$$S_{M_o} \leq \left(\frac{\text{MAX}}{\sqrt{\lambda}}\,\frac{e^{-M_o^2}}{2M_o}\right)^{\rho/(1+\rho)} \int_0^\infty re^{[-r^2/(1+\rho)]}\,dr = \left(\frac{\text{MAX}}{\sqrt{\lambda}}\,\frac{e^{-M_o^2}}{2M_o}\right)^{\rho/(1+\rho)} \cdot \frac{1+\rho}{2}$$

and it is clear that this last quantity can be made arbitrarily small by an appropriate choice of M_o.

Now if for each r in the range $[0, N]$ we let $M(r) = \dfrac{M_o + r}{\sqrt{\lambda}}$, then

$$S = \left(\frac{MAX}{\sqrt{\lambda}}\right)^{\rho/(1+\rho)} \int_0^\infty re^{-r^2}\left[e^{r^2}\int_{(\sqrt{\lambda}\,M(r)-r)}^\infty e^{-z^2}\,dz\right]^{\rho/(1+\rho)} dr = S_{M_o},$$

since $M_o = \sqrt{\lambda}\,M(r) - r$.

Therefore $R < S < \epsilon$, and we have established that $|E - A| < \epsilon$.

The problem is now in a form that is suitable for the application of numerical quadrature techniques requiring finite limits.

Elaine S. Brown