Number Crunching with 8087 FQUANs: The Mie Equations

Ferren MacIntyre

Center for Atmospheric-Chemistry Studies
Graduate School of Oceanography
University of Rhode Island
Narragansett, RI 02882-1197

Abstract

By long-standing tradition, FORTH uses scaled-integer arithmetic in preference to floating-point operations, both because of its origin in the integer-rich world of process control and because of the inefficiencies of floating-point operations. However, scaling requires advance information on magnitudes, which is not always easily available in scientific calculations. In addition, the appearance of the 8087 numerical co-processor has removed the stigma of inefficiency, and we have reached the point anticipated by Charles Moore, at which it becomes preferable to use floating-point operations.

Taking the IBM PC BIOS software single-precision routines as baseline for the heavily numerical Mie equations, the 8087 is 115 times faster. Precision increased from one significant figure with BIOS (round-off error dominates in the recursive claculations) to agreement with published 6-figure values with the 8087.

Because stack operations are fast on the 8087, while 8-byte stores and fetches are relatively slow, it pays to keep operands on the 80-bit-wide, 8-word-deep 8087 stack. It is, for instance, possible to replace two complex numbers with their sum and difference, without leaving the stack.

The net result is a Mie-equation algorithm which produces a result in 15 minutes, compared to the 24-hour turnaround for a 1-second CRAY-1 background run.

Background

The immediate problem which led to the algorithm to be discussed was the observation of brightly colored small air bubbles in water, (in Duncan Blanchard's laboratory at SUNY Albany) (Struthwolf 1983, MacIntyre et al. 1985). Here, a beam of high-intensity light was scattered by the bubbles, and according to the description, both spectral and non-spectral colors were seen, often quite bright, but changing slowly with time (as the bubbles dissolved). Occasionally, alternation of red and green colors as a function of scattering angle was observed. Since exactly the same phenomena are found in interference colors from thin films, it seemed probable that the colors should be predictable by Mie light-scattering theory. This in turn might have the added advantage of providing the diameter of the bubbles, which is difficult to measure by other techniques.

Unfortunately, there were no financial resources for tackling the Mie equations ([1] and [2] below) on a suitable mainframe computer. These equations, which are a solution to

Maxwell's electromagnetic equations for two materials of differing complex dielectric coefficient, are widely recognized as time consuming, and serious optical physicists preferentially work on super-computers such as the CRAY-1.

The time consumption follows from the need for four nested loops. The innermost loop computes the terms of the "infinite" series which are the solution to the scattering equations for a given wavelength, diameter, and scattering angle. The second loop sums these terms into the series. But one is almost always interested in a range of diameters, wavelengths, and angles, and while diameter and wavelength can be combined into the Mie parameter

x = wavelength/circumference

this set of parameters still creates two additional loops. Time increases rapidly as the Mie parameter increases, since the "infinite" series get longer. In addition, the number of oscillations in the answer increases as x increases, so that more points must be computed to draw a smooth intensity curve.

The older literature of light scattering is replete with inadequate solutions to the Mie equations. Frequently, too few calculations were made, so that details of the rapidly oscillating intensity are missed. (For values of x as low as 3.6, some of the minima of van de Hulst's (1957) graphs are too shallow by 3 orders of magnitude.) And there are more recent instances in which published results are totally incorrect over some portion of the range of interest, usually because an otherwise well behaved algorithm has hit a region of instability from which it later recovers.

As a result, considerable ingenuity has been applied to the problem of optimizing Mie algorithms. The most recent algorithm is Wiscombe's (1979, 1980) FORTRAN code for the CRAY-1 at the National Center for Atmospheric Research (NCAR, Boulder, CO), which is carefully structured to minimize computation time, in both vectorized "MIEV1" and non-vectorized "MIEV0" versions.

One hesitates to tackle programs of such sophistication on "home" computers like the IBM PC. Furthermore, in 1983, FORTH was still primarily an integer-arithmetic language. Despite many successes with high-precision calculations in which variables remained within relatively small ranges—which makes scaling feasible—FORTH was not felt to be appropriate for number crunching in problems with exponential ranges. Yet the PC is bigger, faster, and smarter than the IBM 704, on which I first ran the Mie equations, so it seemed a worthwhile challenge to try the Mie equations on the PC, and the price was right.

The Structure of the Mie Equations

Details of the Mie equations may be found in van de Hulst (1957), but they will be sketched here to indicate the problems involved. Since bubbles are spherical, it should be no surprise to find that Ricatti-Bessel functions appear in the solutions, given by

$S_n(x) = xj_n(x)$
$C_n(x) = -xn_n(x)$

where $j_n(x)$ and $n_n(x)$ are the spherical Bessel functions of order $n+1/2$. The symbols come from the fact that $S_0(x) = \sin x$ and $C_0(x) = \cos x$. (In the listing, these are called PSX and CHI, following van de Hulst.)

The actual functions used in the Mie results are combinations of S_n and C_n and their derivatives with respect to x, S′ and C′. The refractive index (dielectric coefficient) used here is in fact the ratio of the actual indices,

$m = m_1/m_2$

and is less than unity when, as here, the index of the sphere, $m_1 = 1$, is smaller than that of the surroundings, $m_2 = 4/3$. (Actually, m_2 is a weak function of the wavelength of light, and was calculated as needed.) Taking advantage of a simplification given by van de Hulst (1957) for real m, (i.e., no absorption—a valid approximation for visible wavelengths over the path lengths of the original experiment) we can write:

$$-\tan \alpha_n = [S'_n(mx)S_n(x) - m\, S_n(mx)S'_n(x)]/[S'_n(mx)C_n(x) - m\, S_n(mx)C'_n(x)]$$

$$-\tan \beta_n = [m\, S'_n(mx)S_n(x) - S_n(mx)S'_n(x)]/[m\, S'_n(mx)C_n(x) - S_n(mx)C'_n(x)]$$

These are actually computed recursively by the method of Dave (1969) as shown in Figure 2., (175:6) (Block:Line). Then

$$a_n = \tan \alpha_n/(\tan \alpha_n - i)$$

$$b_n = \tan \beta_n/(\tan \beta_n - i),$$

where i is $(-1)^{1/2}$.

The angular dependence of the scattered light is given by derivatives of Lengendre polynomials P_n^m,

$$\pi_n(\cos\theta) = dP_n(\cos\theta)/d\cos\theta$$

$$\tau_n(\cos\theta) = -\sin\theta\, dP_n^1(\cos\theta)/d\cos\theta,$$

which are again computed recursively by Wiscombe's algorithm (Block 177).

The complete solution for spherical symmetry is then a pair of "infinite" sums (176, 177:12–14)

$$S_1 = \sum_{n=1}^{N} [(2n+1)/n(n+1)]\{a_n\pi_n + b_n\tau_n\} \qquad [1]$$

$$S_2 = \sum_{n=1}^{N} [(2n+1)/n(n+1)]\{b_n\pi_n + a_n\tau_n\} \qquad [2]$$

where S_1 and S_2 refer to polarizations with electric vector perpendicular and parallel to the plane of scattering, respectively.

Note that although m itself is real, a_n and b_n, and therefore S_1 and S_2, are complex numbers. The polarized scattered intensities—whose calculation is the whole point of the exercise—are then real numbers given by

$$I_1 = |S_1(\theta)|^2$$

$$I_2 = |S_2(\theta)|^2$$

Wiscombe (1979) devoted much care to minimizing the number of arithmetic operations in the innermost loop which computes the sums, arriving at 7 multiplications and 4 additions. This structure transferred into FORTH more or less intact, although, because scattering at constant diameter and constant angle were both required, two versions of the inner loop were written, one somewhat more efficient than the other.

The key to successful calculation, however, lies in knowing when to terminate the infinite series for S_1 and S_2. If one stops too soon, the answers are inaccurate, but if one

continues too long, the approach of the denominators in a_n and b_n toward zero leads to numerical instability no matter what the word length. Wiscombe's most useful contribution was a simple equation giving the required number of terms N, which for our purposes may be written:

$$N = 4.05x^{1/3} + x + 2.$$

Although Wiscombe emphasized that he had tested this only for $m>1$, it appears to work equally well for m near 0.75.

Feeling a need for more check data than the literature provides, I attempted to put a FORTRAN MIEV0 into the campus mainframe. Alas, not all FORTRANs are alike, and after two weeks I abandoned this approach, and returned to FORTH. It is not necessarily inefficient to translate a FORTRAN program into FORTH, although I find that it requires several iterations to produce stylistically acceptable—rather than merely workable—FORTH code when starting from a FORTRAN-like precursor.

Implementing the Mie Equations in MMSFORTH

When I started on this problem, the 8087 was not yet available. The first calculations used the ordinary single-precision (32-bit) floating-point data types, 2CONSTANT and 2VARIABLE, and simply called the existing software floating-point-arithmetic routines in IBM BASIC ROM. The limited precision proved inadequate to cope with the growth of round-off error during recursions, and the approach was generally slow, inaccurate, and unsatisfactory.

One option, firmly in the FORTH tradition, was to use scaled-integer arithmetic. However, scaling becomes impractical unless the magnitude of the calculated numbers can be predicted with some assurance. Consider the recursion in Fig. 1 at 173:7, which has

$$A_n = n/y - 1/(n/y + A_{n-1})$$

In the simplest case we can reduce this to

$$A = a - 1/a,$$

and suppose A and a of comparable magnitude, with $a<1$ having k digits to the right of the decimal point. To preserve them, we scale the entire equation by 10^k, obtaining

$$A.10^k = a.10^k - 10^k/a.$$

But now we must multiply the top and bottom of the fraction by 10^k to make the a in the denominator an integer also, giving

$$A.10^k = a.10^k - 10^{2k}/a.10^k.$$

Choosing 2k to approach machine capacity still only gives us k digits in A. It is for this reason that FORTH includes the mixed-precision operator */, which maintains a double-precision intermediate. However, experiments with */ lead to results like the following: For a = 0.2468 and A = -3.8051, the closest approach is

```
247  1000  10000  2468  */  -  =  -3806
```

Figure 1. At (173:7) (Block:Line) is the central equation for recursively calculating key coefficients in the Mie equations. It is this equation which offers the first difficulties in conversion to scaled integers, as explained in the text.

Nonstandard words abound in the 8087 environment. Operators preceded by "F" and "CP" do the expected thing, but to floating-point and complex numbers on the 8087 stack respectively. >87 moves an integer from the normal stack to the 8087 stack as a floating-point number. 1/X gets the reciprocal of the top of the 8087 stack. The structure xxx I IS (A) stores xxx into the I-th location of FQUAN array (A).

```
Block 173 [173 :0]

 0 ( 840815 FM Mie Intensity          6/15 Logarithmic derivative )
 1
 2 ( : A?      I' . FDUP F. 2 COLS ; )
 3 ( Compute [A] {Y} by downward recursion from A {N'}=0 where
 4 N' is taken to be 5/3 Nmax, {because 1.5 Nmax gives the same
 5 result}, and n [A] is Lentz's A-sub-n, psi'/psi)
 6 ( This is the equation that won't go in scaled integer)
 7 : W22    >87 Y F/ FDUP    FROT    F+ 1/X F- ;        (Wiscombe 22)
 8
 9 : ADN    ZERO ( = A{N'} )     N 1+     N 5 * 3 / ( 5/3 N to N)
10     DO I W22 ( A? ) -1 +LOOP FDUP    N IS [A]     ( CR)
11     ( A{N} ->)     2 n
12     DO I W22 ( A? ) FDUP I 1- IS [A] -1 +LOOP FDROP ;
13                                                        -->
14
15
```

Unhappily, writing 2468 10000 . . . , in an attempt to obtain one more digit, leads not to -38051, but to 10404. One must monitor such calculations carefully to avoid overflow, and adjust scaling as required. This seems scarcely practical when successive terms in a recursion may change by several orders of magnitude.

A second option would have been quadruple-precision 4CONSTANT and 4VARIABLE (64-bit) data types and an extended-precision version of */. But this would have been considerably slower, and runs were already taking all day. As it happened, the 8087 became available about this time, so this option was not pursued. Instead, I prevailed upon Tom Dowling of Miller Microcomputer Services to write a driver for the 8087.

The hardware and software arrived simultaneously and with the usual beta-test-site documentation ("Throw switch 2. Load block 20"). Two days later the program had been converted to work with the 8087, and at this time I ran my one internal benchmark: A set of parameters which had taken 9:30 hours now ran in 5:07 minutes, for a 115-fold speed increase. In BASIC, this calculation would have required approximately a week. Equally useful was the increase in precision, from one significant figure to 6 or more (since published test values were reported only to 6 figures).

This result confirms the current folk wisdom: the 8087 is the cheapest computer power on the market.

RPN vs. Algebraic Notation

Figure 2 shows the RPN version of the inner loop which sums S+ = (S1+S2)/2 and S- = (S1-S2)/2. After several years of practice with HP calculators and FORTH, I am fairly adept at translating algebraic notation into RPN. I find that this is not a reversible skill, and that each time I try to unravel block 175, to reconstruct the algebra, I obtain a different answer, even though I left what I thought was an abundance of notes about the contents of the stack at various points. Perhaps when RPN is initially taught in the classroom, some attention should be paid to developing skills in both directions.

Proponents of RPN might enjoy recreating the algebraic equations underlying blocks 174, 175, and 177.

Figure 2. Three blocks from the core of the Mie-equation program illustrating (1) the sort of notes which seem to be required to keep RPN and algebraic equations related in any meaningful way (174:1-4,6,8-10,13,14; 175:1-4,8; 177:1,3-9,11,14); and (2) the structure of the inner loop in which the machine spends most of its time (SSV, 177:2).

FINIT initializes the 8087 stack, -F- is a single 8087 instruction which does (a b -> b-a), F+- does (a b -> a-b a+b), and FC* does (c a bi -> ac bci). MAG^2 does (a bi -> $a^2 + b^2$).

```
Block 174 [174 :0]
 0 ( 16\08\83 FM Mie Intensity          7/15 )
 1 ( Dave JV, 1969, IBM J. Res. Dev. 13,302-313; in TN140 as eq 16
 2  except that CHI is the real-valued part of ZETA. The imaginary
 3  part of an and bn comes from the transformation given by
 4  vdH p 135 for real refractive index.)
 5
 6 : DAVE2-3              ( A/or*m I -> an or bn )                    (Wiscombe 16)
 7          >87            X           F/          F+ FDUP    PSX F*  PSX-1 F-
 8 ( A/*m + n/z     psi - psi-1                       FSWAP  CHI F*  CHI-1 F- F/
 9 ( *{2n+1}/n{n+1} now)                              FDUP   NN  F*
10 ( -tan{alpha|beta} for real m)                  % 0 FROT   ONE CP/ ;
11
12
13 ( On 1st pass, PSX is psi-1, PSX-1 is psi-0.     a&b then puts
14 psi-0 into PSX-2 for n=2 pass.)
15

Block 175 [175 :0]
 0 ( 29\06\84 FM Mie Intensity          8/15 van de Hulst 1957 p 135 )
 1   ( an     and     bn     are the complex-valued Mie coefficients.
 2     an = {-tan alpha}/{-tan alpha + i } * {2n+1}/2n{n+1}
 3     bn = {-tan beta }/{-tan beta  + i } * {2n+1}/2n{n+1}
 4     a+b = an + bn,     a-b = an - bn )
 5 : a&b      FINIT
 6   I' [A]     m F/          I' DAVE2-3 ( an      )
 7   I' [A]     m F*          I' DAVE2-3 ( an bn )     CP+-   I' IS a+b I' IS a-b
 8     ( Shift Ricatti-Bessel functions for next recursion)
 9          PSX-1 IS PSX-2       PSX IS PSX-1
10          CHI-1 IS CHI-2       CHI IS CHI-1     FINIT ;
11
12
13
14
15
```

```
Block 177 [177 :0]

 0 ( 02\08\83 FM Mie Intensity          10/15 Wiscombe p 1507 Sec. III )
 1 ( LCS=0: Scattering functions     S+ and S- for varying theta)
 2 : SSV        ZEROP ZEROS
 3       N 1+ 1 DO ( Legendre polynomial recursion) ( Stack:)
 4                    PIn FDUP MU F*     FDUP            ( pn s s)
 5                             PI- F-                    ( pn s t)
 6                         I 1+ >87 F*                   ( pn s {n+1} t)
 7                    FOVER FOVER -F-    FROT FROT      ( pn tau s {n+1} t)
 8                      I >87     F/ F+    IS PI+        ( pn tau)
 9                                F+-                    ( pn-tau     pn+tau)
10
11 ( Sum scattering functions {pi+tau} {an+bn} and {pi-tau} {an-bn})
12                          ( I p+t       ) I a+b    FC*     S+ CP+    IS S+
13                          ( I p-t       ) I a-b    FC*     S- CP+    IS S-
14    ( Shift Legendre polynomial)  PIn     IS PI-
15                                  PI+     IS PIn            LOOP ;            -->
```

FQUANS and Special Operators

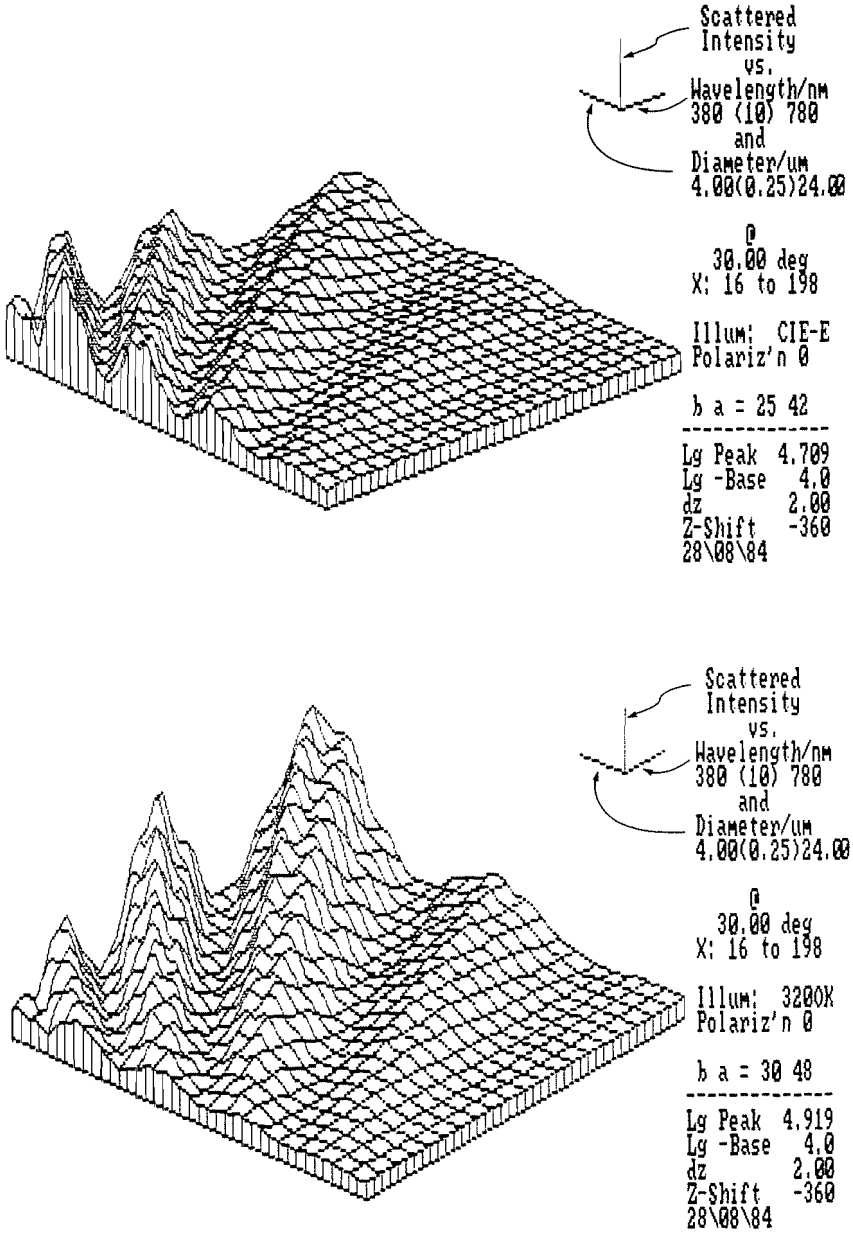
The QUAN, (Rosen 1982, Dowling 1983) is a data type with three code-fields specifying its fetch routine, its store routine, and its location. Called by name, and stored by IS (as in 17 IS AQUAN), it eliminates the ' and @ that accompany CONSTANT and VARIABLE, and—more important—the necessity of remembering how one has defined data. The 32-bit FQUAN and 64-bit DFQUAN are similar floating-point data types, the latter approaching the 80-bit precision of the 8087 itself. Slight extensions give us the CPQUAN, for storing complex numbers, and array-defining parent words such as FQARRAY, DFQARRAY, CPQARRAY and their two-dimensional counterparts.

Since two arrays of 40 × 80 × 8 bytes or 25Kb each (1k=1000, 1K=1024) may be wanted in these calculations, it is essential to put the data (minus headers and code-fields) into memory above FFFF. For such purposes, Dowling wrote a set of long-address array-defining words LFQARRAY, LDFQARRAY, LCPQARRAY, L2FQARRAY, L2DFQARRAY, and L2CPQARRAY, which, at the cost of a 23% increase in overhead on fetches and stores, allow one to use upper memory for data storage. Most of the features are proprietary in their optimized form, and available on the MMSFORTH Utilities disc.

Two complex numbers on the 8087 stack use 4 of the 8 available stack registers. A recurring need in the program was to have available the results of operations on complex numbers, rather than the numbers themselves, and some attention was paid to creating operators which would achieve such goals without the need for time-consuming storage of intermediate results. As an example, one resulting word, CP+-, which returns the sum and difference of two complex numbers on the 8087 stack, is

```
              ( a bi c di -> a-c {b-d}i a+c {b+d} i )
CODE CP+- ST(1) FLD ST ST(4) FSUBR ST(1) FLD ST ST(4) FSUBR
          ST(4) FXCH  ST(2) FADDP ST(4) FXCH  ST(4) FADDP
          NEXT
```

Figure 3. Intensity of unpolarized light scattered at 30 degrees, as a function of wavelength and bubble diameter. Drawings by screendump of MMSFORTH's TGRAPH, extended a bit for the occasion. Above, with spectrally flat illumination; below, with a 3200 K blackbody, approximating a slide-projector bulb.



It was particularly gratifying to be able to produce such words using FORTH's trial-and-error capability, since despite several inquiries to Intel I had been unable to obtain anything more helpful than a bare list of 8087 mnemonics, with no accompanying description of their operation or operands!

Results

Typical of the data generated in such a run is Fig. 3, which shows the scattered intensity at 25 degrees for all visible wavelengths and a range of diameters from 4 to 24 micrometers.

The eye/brain, of course, integrates the intensities it receives from all wavelengths and reports a single color to the user. Converting the spectra of Fig. 3 into perceived colors is a problem in the "psychophysics" of color, the method being that each spectrum (say, along a given diameter in Fig. 3) is multiplied wavelength by wavelength by three physiological response functions of the mythical "standard observer", to yield three perceived intensities, red X, green Y, and blue Z. To show these three coordinates conveniently in two dimensions, they are normalized by dividing by their sum,

$$x = X/(X + Y + Z)$$
$$y = Y/(X + Y + Z)$$
$$z = Z/(X + Y + Z).$$

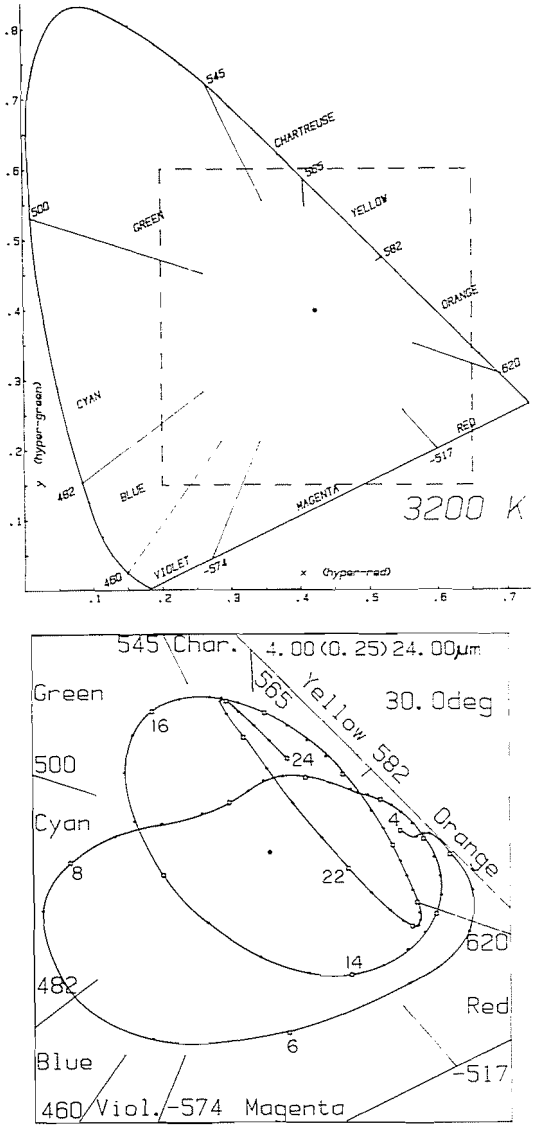
The z coordinate is now redundant because x+y+z = 1, but x and y (essentially red and green, except that unity on these axes represents a color of greater spectral purity than the eye can perceive) can now be plotted. In this process the total intensity is lost (the intensity point [X,Y,Z] has been projected downward onto the x,y plane), but the color now lies within the bounds of the "chromaticity diagram" of Fig. 4 (see, e.g., Wyszecki and Styles 1967). The most accessible color picture of a chromaticity diagram is the cover of "Kodak Filters" (Anonymous 1978), $4 at most camera stores.

The color of the illuminant is perceived as achromatic by the eye over a broad range of actual colors: Fig. 4 is drawn for a 3200°K blackbody illuminant (i.e., a slide-projector bulb), which is normally *perceived* as white in use, although it is in fact much redder than, say, noon skylight. The curve surrounding this intersection is the locus of perceived color as one watches a bubble whose diameter is growing from 4 to 24 micrometers, at a scattering angle of 25 degrees.

Colors near the achromatic point are bright but washed out and nearly white; colors at the circumscribing locus are spectrally pure but of low intensity. Colors midway along the radials are less spectrally pure, but brighter, and many of the colors scattered by bubbles lie near the point of maximum brilliance. In fact, colors refracted and diffracted by bubbles are intrinsically much brighter than rainbows reflected and dispersed from drops. It is this fact which makes it possible to "see" a 2-μm bubble, which is well below the resolution of the eye to see as a circular object.

The bubble changes color very rapidly for small diameters and small scattering angles. Sufficiently small bubbles are white, but by 4 μm have become a vivid orange. Further growth produces nonspectral magentas (which means only that they are relatively deficient in yellow and green wavelengths), then blues and greens back to yellow and orange again by the time the bubble has grown to 10 μm. Further growth to 18 μm swings the locus through all colors once again. However, growth beyond 18 μm produces the characteristic red-green oscillation noted in the original data (Struthwolf 1983). Growth beyond that covered here continues this oscillation with decreasing amplitude and a reapproach to the white point. In general, the colors of bubbles correspond to the colors of films an order of magnitude thinner than the bubble diameter. A particularly useful colored-pencil sketch of the reason for these color sequences is given by Boys (1911).

Figure 4. On the chromaticity diagram, all visible colors lie inside the curving spectral locus. The central point is white. The x and y coordinates of colors can be computed from their spectral distribution. Plots by MMSFORTH's FortHPlot driving the HP 7470A plotter. Above, the basic diagram showing the dashed window expanded below. Below, peregrinations of the chromaticity locus as bubble diameter changes.



Conclusions

The end result of this effort is a light-scattering program which (on the basis of some limited benchmarking) competes favorably with MIEV0 run on the CRAY-1. The CRAY-1 is about 1000 times faster (test runs which the CRAY-1 does in 1 second require about 15 minutes on the PC), but, as a reviewer pointed out, the CRAY costs 2000 times as much, giving the PC a 2-to-1 performance/price advantage when large memory is not an issue. But watching my peers use the CRAY by remote access reveals that the turnaround time for

1-second background jobs is 24 hours, in which time I can produce a goodly number of 15-minute runs on the IBM PC!

I wrote the Mie equations from the textbooks, not from MIEV0, so they do somewhat different things, and it is difficult to compare program sizes. However, I find that when I translate adequately documented FORTRAN into adequately documented FORTH, the number of lines does not change.

An additional advantage of working close to home is the ability to design and optimize output graphics easily. I know of few occupations less productive than trying to use a graphics program in batch mode. In addition to the "fishnet" of Figure 3 and chromaticity locus of Figure 4, a most useful tool for detecting errors was a quick-and-dirty character-mapped contour plot which displayed results as fast as it could fill the screen.

From the optical point of view, it appears that simple, colloquial, descriptions of color seen at three scattering angles, say 25, 35, and 45 degrees, will enable one to determine the diameter of a small bubble to an accuracy of 0.5 μm over the range 2-25 μm.

From the programmer's point of view, it appears that FORTH can hold its own as a number-crunching language when teamed with the 8087. I have no reservations about recommending this particular combination (MMSFORTH and the 8087) for serious scientific programming at minimum cost.

Acknowledgements

This work was inadvertently supported by NSF grant OCE 81-17849. I thank Tom Dowling of Miller Microcomputer Services for writing fast 8087 code, and Jim Gerow for explaining what Dowling intended it to do. Philip Marston of the University of Washington (Pullman) and Warren Wiscombe shared helpful experiences with the Mie equations.

References

Anonymous, 1978. "Kodak Filters for Scientific and Technical Uses." *Kodak Pub. No B-3.* (Rochester, NY).

Dave, J.V., 1969. Scattering of electromagnetic radiation by a large absorbing sphere. *IBM J. Res. Dev.* 13:302–313.

Dowling, T., 1983. The QUAN concept expanded. *1983 Rochester Forth Applications Conf.* pp. 89–92.

MacIntyre, F., M. Struthwolf and D.C. Blanchard, 1984. Color effects in the scattering of light by small air bubbles in water. *J. Opt. Soc. Amer.* (submitted).

Rosen, E., 1982. QUAN and VECT —High-speed, low-memory-consumption structures. *Proc. Fourth FORML Conf.*

Struthwolf, M., 1983. Behavior of air bubbles <400 μm diameter in and at the surface of seawater and distilled water. Master's thesis, State University of New York at Albany.

Van de Hulst, H.C., 1957. *Light Scattering by Small Particles.* (Wiley-Interscience, NY).

Wiscombe, W.J., 1979. Mie scattering calculations: Advances in technique and fast, vector-speed computer codes. NCAR/TN-140+STR.

Wiscombe, W.J., 1980. Improved Mie scattering algorithms. *Appl. Optics* 19:1505–1509.

Wyszecki, G. and W.S. Stiles, 1967. *Color Science.* (Wiley-Interscience, NY).

Dr. MacIntyre is a high school drop-out with a PhD in Physical Chemistry from MIT. He is currently a Research Professor in the Graduate School of Oceanography, at the University of Rhode Island. He is interested in the top micron of the ocean and its subset: bubbles and aerosol.

Manuscript received October 1984.