Compiler used: [GDB online Debugger | Compiler - Code, Compile, Run, Debug online C, C++]()

Code copied from the textbook:

```
      PROGRAM CALLBH (INPUT=TTY,OUTPUT=TTY,TAPE5=TTY)
C*************************************************************************
C     CALLBH CALCULATES THE SIZE PARAMETER (X) AND RELATIVE
C     REFRACTIVE INDEX (REFREL) FOR A GIVEN SPHERE REFRACTIVE
C     INDEX, MEDIUM REFRACTIVE INDEX, RADIUS, AND FREE SPACE
C     WAVELENGTH.  IT THEN CALLS BHMIE, THE SUBROUTINE THAT COMPUTES
C     AMPLITUDE SCATTERING MATRIX ELEMENTS AND EFFICIENCIES
C*************************************************************************
      COMPLEX REFREL,S1(200),S2(200)
      WRITE (5,11)
C*************************************************************************
C     REFMED = (REAL) REFRACTIVE INDEX OF SURROUNDING MEDIUM
C*************************************************************************
      REFMED=1.0
C*************************************************************************
C     REFRACTIVE INDEX OF SPHERE = REFRE + I*REFIM
C*************************************************************************
      REFRE=1.55
      REFIM=0.0
      REFREL=CMPLX(REFRE,REFIM)/REFMED
      WRITE (5,12) REFMED,REFRE,REFIM
C*************************************************************************
C     RADIUS (RAD) AND WAVELENGTH (WAVEL) SAME UNITS
C*************************************************************************
      RAD=.525
      WAVEL=.6328
      X=2.*3.14159265*RAD*REFMED/WAVEL
      WRITE (5,13) RAD,WAVEL
      WRITE (5,14) X
C*************************************************************************
C     NANG = NUMBER OF ANGLES BETWEEN 0 AND 90 DEGREES
C     MATRIX ELEMENTS CALCULATED AT 2*NANG - 1 ANGLES
C     INCLUDING 0, 90, AND 180 DEGREES
C*************************************************************************
      NANG=11
      DANG=1.570796327/FLOAT(NANG-1)
      CALL BHMIE(X,REFREL,NANG,S1,S2,QEXT,QSCA,QBACK)
```

```
      WRITE (5,65) QSCA,QEXT,QBACK
      WRITE (5,17)
C**********************************************************************
C    S33 AND S34 MATRIX ELEMENTS NORMALIZED BY S11.
C    S11 IS NORMALIZED TO 1.0 IN THE FORWARD DIRECTION
C    POL=DEGREE OF POLARIZATION (INCIDENT UNPOLARIZED LIGHT)
C**********************************************************************
      S11NOR=0.5*(CABS(S2(1))**2+CABS(S1(1))**2)
      NAN=2*NANG-1
      DO 355 J=1,NAN
      AJ=J
      S11=0.5*CABS(S2(J))*CABS(S2(J))
      S11=S11+0.5*CABS(S1(J))*CABS(S1(J))
      S12=0.5*CABS(S2(J))*CABS(S2(J))
      S12=S12-0.5*CABS(S1(J))*CABS(S1(J))
      POL=-S12/S11
      S33=REAL(S2(J)*CONJG(S1(J)))
      S33=S33/S11
      S34=AIMAG(S2(J)*CONJG(S1(J)))
      S34=S34/S11
      S11=S11/S11NOR
      ANG=DANG*(AJ-1.)*57.2958
  355 WRITE (5,75) ANG,S11,POL,S33,S34
   65 FORMAT (//,1X,"QSCA= ",E13.6,3X,"QEXT = ",E13.6,3X,
     2"QBACK = ",E13.6)
   75 FORMAT (1X,F6.2,2X,E13.6,2X,E13.6,2X,E13.6,2X,E13.6)
   11 FORMAT (/"SPHERE SCATTERING PROGRAM"//)
   12 FORMAT (5X,"REFMED = ",F8.4,3X,"REFRE =",E14.6,3X,
     3"REFIM = ",E14.6)
   13 FORMAT (5X,"SPHERE RADIUS = ",F7.3,3X,"WAVELENGTH = ",F7.4)
   14 FORMAT (5X,"SIZE PARAMETER =",F8.3/)
   17 FORMAT (//,2X,"ANGLE",7X,"S11",13X,"POL",13X,"S33",13X,"S34"//)
      STOP
      END
C**********************************************************************
C    SUBROUTINE BHMIE CALCULATES AMPLITUDE SCATTERING MATRIX
C    ELEMENTS AND EFFICIENCIES FOR EXTINCTION, TOTAL SCATTERING
C    AND BACKSCATTERING FOR A GIVEN SIZE PARAMETER AND
C    RELATIVE REFRACTIVE INDEX
C**********************************************************************
```

```
      SUBROUTINE BHMIE (X,REFREL,NANG,S1,S2,QEXT,QSCA,QBACK)
      DIMENSION AMU(100),THETA(100),PI(100),TAU(100),PIO(100),PI1(100)
      COMPLEX D(3000),Y,REFREL,XI,XI0,XI1,AN,BN,S1(200),S2(200)
      DOUBLE PRECISION PSI0,PSI1,PSI,DN,DX
      DX=X
      Y=X*REFREL
C***********************************************************************
C     SERIES TERMINATED AFTER NSTOP TERMS
C***********************************************************************
      XSTOP=X+4.*X**.3333+2.0
      NSTOP=XSTOP
      YMOD=CABS(Y)
      NMX=AMAX1(XSTOP,YMOD)+15
      DANG=1.570796327/FLOAT(NANG-1)
      DO 555 J=1,NANG
      THETA(J)=(FLOAT(J)-1.)*DANG
  555 AMU(J)=COS(THETA(J))
C***********************************************************************
C     LOGARITHMIC DERIVATIVE D(J) CALCULATED BY DOWNWARD
C     RECURRENCE BEGINNING WITH INITIAL VALUE 0.0 + I*0.0
C     AT J = NMX
C***********************************************************************
      D(NMX)=CMPLX(0.0,0.0)
      NN=NMX-1
      DO 120 N=1,NN
      RN=NMX-N+1
  120 D(NMX-N)=(RN/Y)-(1./(D(NMX-N+1)+RN/Y))
      DO 666 J=1,NANG
      PIO(J)=0.0
  666 PI1(J)=1.0
      NN=2*NANG-1
      DO 777 J=1,NN
      S1(J)=CMPLX(0.0,0.0)
  777 S2(J)=CMPLX(0.0,0.0)
C***********************************************************************
C     RICCATI-BESSEL FUNCTIONS WITH REAL ARGUMENT X
C     CALCULATED BY UPWARD RECURRENCE
C***********************************************************************
      PSI0=DCOS(DX)
      PSI1=DSIN(DX)
```

```
    CHI0=-SIN(X)
    CHI1=COS(X)
    APSI0=PSI0
    APSI1=PSI1
    XI0=CMPLX(APSI0,-CHI0)
    XI1=CMPLX(APSI1,-CHI1)
    QSCA=0.0
    N=1
200 DN=N
    RN=N
    FN=(2.*RN+1.)/(RN*(RN+1.))
    PSI=(2.*DN-1.)*PSI1/DX-PSI0
    APSI=PSI
    CHI=(2.*RN-1.)*CHI1/X - CHI0
    XI=CMPLX(APSI,-CHI)
    AN=(D(N)/REFREL+RN/X)*APSI - APSI1
    AN=AN/((D(N)/REFREL+RN/X)*XI-XI1)
    BN=(REFREL*D(N)+RN/X)*APSI - APSI1
    BN=BN/((REFREL*D(N)+RN/X)*XI - XI1)
    QSCA=QSCA+(2.*RN+1.)*(CABS(AN)*CABS(AN)+CABS(BN)*CABS(BN))
    DO 789 J=1,NANG
    JJ=2*NANG-J
    PI(J)=PI1(J)
    TAU(J)=RN*AMU(J)*PI(J) - (RN+1.)*PIO(J)
    P=(-1.)**(N-1)
    S1(J)=S1(J)+FN*(AN*PI(J)+BN*TAU(J))
    T=(-1.)**N
    S2(J)=S2(J)+FN*(AN*TAU(J)+BN*PI(J))
    IF(J.EQ.JJ) GO TO 789
    S1(JJ)=S1(JJ) + FN*(AN*PI(J)*P+BN*TAU(J)*T)
    S2(JJ)=S2(JJ)+FN*(AN*TAU(J)*T+BN*PI(J)*P)
789 CONTINUE
    PSI0=PSI1
    PSI1=PSI
    APSI1=PSI1
    CHI0=CHI1
    CHI1=CHI
    XI1=CMPLX(APSI1,-CHI1)
    N=N+1
    RN=N
```

```
   DO 999 J=1,NANG
   PI1(J)=((2.*RN-1.)/(RN-1.))*AMU(J)*PI(J)
   PI1(J)=PI1(J)-RN*PIO(J)/(RN-1.)
999 PIO(J)=PI(J)
   IF (N-1-NSTOP) 200,300,300
300 QSCA=(2./(X*X))*QSCA
   QEXT=(4./(X*X))*REAL(S1(1))
   QBACK=(4./(X*X))*CABS(S1(2*NANG-1))*CABS(S1(2*NANG-1))
   RETURN
   END
```

Running the BHMIE Code in GDB:

1. Open a new project in GDB
2. In the "Language" Tab, select Fortran
3. Click the settings icon and select "Extra Compiler Tags", then paste "-ffixed-form -std=legacy -ffixed-line-length-none" in the window that appears. This sets the program file to be in Fortran77, despite being saved as a Fortran95 file.
4. Paste the code directly from the textbook.
5. The textbook states "the first statement in each calling program is not executable by many Fortran compilers; also, input, output, and format statements may have to be changed" (page 475, pdf page 482), which hold true in this one. The following edits have to be made in order for the code to compile correctly:
   a. In line 1, delete "(INPUT=TTY,OUTPUT=TTY,TAPE5=TTY)", so the line just reads "PROGRAM CALLBH"
   b. Anywhere a line begins with "WRITE (5", replace it with "write (*"
      i. There is a command for this by using CTRL+F, then searching for "WRITE (5" and clicking the plus arrow underneath the search bar. This will prompt you with a "Replace with" text box that you can type "write (*" into and hit "ALL" on the right to replace every "WRITE (5" with "write (*"
6. With these changes, the output will be the exact same as the one shown in the textbook.

Translating Code from Fortran77 to Python:

I used this website to translate it: [Fortran to Python Converter] (note that you'll have to remove the comments so it doesn't exceed the 4000 character limit since this is the free version)
The output is correct, just not formatted in the same manner as the original output. Changing the numerical values to type float fixed this for me. The modified Python code and its output is shown below.

Python Code:

```
import math

def bhmie(x: float, refrel: complex, nang: int):
    y = x * refrel
    xstop = x + 4.0 * (x ** (1.0 / 3.0)) + 2.0
    nstop = int(xstop)
    ymod = abs(y)
    nmx = int(max(xstop, ymod)) + 15
    dang = (math.pi / 2.0) / float(nang - 1)
    theta = [j * dang for j in range(nang)]
    amu = [math.cos(t) for t in theta]
    d = [0j] * (nmx + 1)
    d[nmx] = 0j
    for n in range(1, nmx):
        rn = float(nmx - n + 1)
        i = nmx - n
        d[i] = (rn / y) - (1.0 / (d[i + 1] + rn / y))
    pio = [0.0] * nang
    pi1 = [1.0] * nang
    nn = 2 * nang - 1
    s1 = [0j] * nn
    s2 = [0j] * nn
    psi0 = math.cos(x)
    psi1 = math.sin(x)
    chi0 = -math.sin(x)
    chi1 =  math.cos(x)
    xi0 = complex(psi0, -chi0)
    xi1 = complex(psi1, -chi1)
    qsca = 0.0
    for n in range(1, nstop + 1):
        dn = float(n)
        rn = dn
        fn = (2.0 * rn + 1.0) / (rn * (rn + 1.0))
        psi = (2.0 * dn - 1.0) * psi1 / x - psi0
        chi = (2.0 * rn - 1.0) * chi1 / x - chi0
        xi = complex(psi, -chi)
        an_num = (d[n] / refrel + rn / x) * psi - psi1
        an_den = (d[n] / refrel + rn / x) * xi  - xi1
```

```
        an = an_num / an_den
        bn_num = (refrel * d[n] + rn / x) * psi - psi1
        bn_den = (refrel * d[n] + rn / x) * xi  - xi1
        bn = bn_num / bn_den
        qsca += (2.0 * rn + 1.0) * (abs(an) ** 2 + abs(bn) ** 2)
        pi = [0.0] * nang
        tau = [0.0] * nang
        p = 1.0 if ((n - 1) % 2 == 0) else -1.0
        t = 1.0 if (n % 2 == 0) else -1.0
        for jF in range(1, nang + 1):
            j = jF - 1
            jjF = 2 * nang - jF
            jj = jjF - 1
            pi[j] = pi1[j]
            tau[j] = rn * amu[j] * pi[j] - (rn + 1.0) * pio[j]
            s1[jF - 1] += fn * (an * pi[j] + bn * tau[j])
            s2[jF - 1] += fn * (an * tau[j] + bn * pi[j])
            if jF != jjF:
                s1[jj] += fn * (an * pi[j] * p + bn * tau[j] * t)
                s2[jj] += fn * (an * tau[j] * t + bn * pi[j] * p)
        psi0, psi1 = psi1, psi
        chi0, chi1 = chi1, chi
        xi1 = complex(psi1, -chi1)
        if n < nstop:
            rn_next = float(n + 1)
            for j in range(nang):
                pi1[j] = ((2.0 * rn_next - 1.0) / (rn_next - 1.0)) * amu[j] * pi[j] \
                        - (rn_next / (rn_next - 1.0)) * pio[j]
                pio[j] = pi[j]
    qsca = (2.0 / (x * x)) * qsca
    qext = (4.0 / (x * x)) * (s1[0].real)
    qback = (4.0 / (x * x)) * (abs(s1[nn - 1]) ** 2)
    return s1, s2, qext, qsca, qback

def main():
    print("\nSPHERE SCATTERING PROGRAM\n")
    refmed = 1.0
    refre = 1.55
    refim = 0.0
    refrel = complex(refre, refim) / refmed
```

```
    print(f"     REFMED = {refmed:8.4f}    REFRE ={refre:14.6E}    REFIM = {refim:14.6E}")
    rad = 0.525
    wavel = 0.6328
    x = 2.0 * math.pi * rad * refmed / wavel
    print(f"     SPHERE RADIUS = {rad:7.3f}    WAVELENGTH = {wavel:7.4f}")
    print(f"     SIZE PARAMETER ={x:8.3f}\n")
    nang = 11
    s1, s2, qext, qsca, qback = bhmie(x, refrel, nang)
    print(f"\n QSCA= {qsca:13.6E}    QEXT = {qext:13.6E}    QBACK = {qback:13.6E}")
    print("\n  ANGLE       S11              POL              S33              S34\n")
    s11nor = 0.5 * (abs(s2[0])**2 + abs(s1[0])**2)
    nn = 2 * nang - 1
    dang = (math.pi / 2.0) / float(nang - 1)
    for j in range(nn):
        s11 = 0.5 * abs(s2[j])**2 + 0.5 * abs(s1[j])**2
        s12 = 0.5 * abs(s2[j])**2 - 0.5 * abs(s1[j])**2
        pol = -s12 / s11
        prod = s2[j] * s1[j].conjugate()
        s33 = (prod.real) / s11
        s34 = (prod.imag) / s11
        s11n = s11 / s11nor
        ang = dang * float(j) * 57.2958
        print(f" {ang:6.2f}  {s11n:13.6E}  {pol:13.6E}  {s33:13.6E}  {s34:13.6E}")

main()
```

Python Output:

```
SPHERE SCATTERING PROGRAM

     REFMED =    1.0000    REFRE =  1.550000E+00    REFIM =    0.000000E+00
     SPHERE RADIUS =    0.525    WAVELENGTH =  0.6328
     SIZE PARAMETER =    5.213


 QSCA=  3.105426E+00    QEXT =  3.105426E+00    QBACK =  2.925341E+00

  ANGLE       S11              POL              S33              S34

   0.00    1.000000E+00  -0.000000E+00    1.000000E+00    0.000000E+00
```

9.00	7.853904E-01	-4.598112E-03	9.994001E-01	3.432611E-02
18.00	3.568971E-01	-4.585405E-02	9.860215E-01	1.601843E-01
27.00	7.661186E-02	-3.647445E-01	8.436025E-01	3.940765E-01
36.00	3.553552E-02	-5.349972E-01	6.869672E-01	-4.917867E-01
45.00	7.018448E-02	9.599526E-03	9.598252E-01	-2.804344E-01
54.00	5.743134E-02	4.779268E-02	9.853711E-01	1.635837E-01
63.00	2.196596E-02	-4.406038E-01	6.480429E-01	6.212155E-01
72.00	1.259589E-02	-8.319957E-01	2.032551E-01	-5.162079E-01
81.00	1.737501E-02	3.416701E-02	7.953537E-01	-6.051819E-01
90.00	1.246008E-02	2.304625E-01	9.374971E-01	2.607419E-01
99.00	6.790928E-03	-7.134719E-01	-7.173975E-03	7.006471E-01
108.00	9.542386E-03	-7.562554E-01	-3.947475E-02	-6.530846E-01
117.00	8.634185E-03	-2.812153E-01	5.362505E-01	-7.958350E-01
126.00	2.274212E-03	-2.396118E-01	9.676018E-01	7.957986E-02
135.00	5.439975E-03	-8.508037E-01	1.875308E-01	-4.908821E-01
144.00	1.602435E-02	-7.063344E-01	4.952544E-01	-5.057813E-01
153.00	1.888524E-02	-8.910812E-01	4.532768E-01	-2.268173E-02
162.00	1.952539E-02	-7.833195E-01	-3.916132E-01	4.827522E-01
171.00	3.016761E-02	-1.961939E-01	-9.620689E-01	1.895556E-01
180.00	3.831891E-02	-0.000000E+00	-1.000000E+00	0.000000E+00