Electronic Supplementary Information (16 pages)

"*Accurate determination of the dielectric parameters of spherical shells in suspension*"
by Niloofar Asgharian and Zoltan A. Schelly

The following two programs are written in C language. Program 1 is used to find the dispersion parameters from the dielectric parameters, and Program 2 is used to find the dielectric parameters from the dispersion parameters. For each program, the input parameters, the actual program, and the output parameters are given sequentially.

Program 1

Input parameters

```
; dielec.in -- input file
; order of variables is fixed in dielec.c
; do not change order.
; numbers must begin with a numerical digit: 0123456789
; for variables that increment, use this format:
; start increment stop
; where start is starting value for the variable
;       increment is the amount to add to the variable each time
;       stop is the last value or upper/lower bound on the variable
;       use increment 0.0 for non-iterating variable

; Ep -- non-iterative variable
        95.7 0.0 0.0
; Kp -- non-iterative variable
        0.0464 0.0 0.0
; Es -- iterative variable
        2.65 0.0 1.95
; Ks -- iterative variable
        1.0e-20 0.0 0.0
; Em -- non-iterative
        78.6 0.0 0.0
; Km -- non-iterative
        7.47e-4 0.0 0.0
; rd -- non-iterative
        724.25 0.0 0.0
; d -- iterative variable
        4.6 0.0 4.71
; p -- iterative variable
        0.201 0.0 0.0
; e0 -- non-iterating variable
        8.85e-12
```

Program 1

```
/* dielec.c */
```

```
/* Program to find dispersion parameters from phase parameters */

#include <stdio.h>
#include <math.h>
#include <string.h>

/* user variables */
/* double uses 8 bytes to store 1 number */

struct inputVar
{
  double start, incr, stop, value;
};

double Ep;
double Kp;
struct inputVar Es;
struct inputVar Ks;
double Em;
double Km;
double rd;
struct inputVar d;
struct inputVar p;
double e0;

/* calculated variables */

double v;
double R1;
double R2;
double R3;
double R4;
double R5;
double R6;
double R7;
double h;
double R;
double B;
double D;
double Eh;
double G;
double S;
double A;
double C;
double E;
double F;
double U;
double El;
double T1;
double T2;
double Tp;
double Tq;
double Dli;
double Dih;
double Dlh;
double Dlhp;
double Kl;
double Dil;
```

```
double Dhi;
double Dhl;
double Kh;
double Dhlp;
double Q;

/* getNextValue() */

int getNextValue(FILE *file, struct inputVar *valuePtr)
{
	int success = 0;
	char line[256];

	while (fgets(line, 255, file) && !isdigit(line[0]));
	if (isdigit(line[0]))
	{
	   sscanf(line, "%le %le %le",
	                &(valuePtr->start),
	                &(valuePtr->incr),
	                &(valuePtr->stop));
	   valuePtr->value = valuePtr->start;
	   success = 1;
	}

	return success;
}


/* readUserVariables() */

int readUserVariables(char *fileName)
{
	int returnValue = 0;
	FILE *inputFile;
	struct inputVar value;

	/* open the file */
	inputFile = fopen(fileName, "r");
	if (inputFile)
	{
	   /* file opened successfully */

	   /* change this section if you need a different
	    * set of input variables
	    */
	   if (getNextValue(inputFile, &value))
	           Ep = value.start;
	   if (getNextValue(inputFile, &value))
	           Kp = value.start;
	   if (getNextValue(inputFile, &value))
	           Es = value;
	   if (getNextValue(inputFile, &value))
	           Ks = value;
	   if (getNextValue(inputFile, &value))
	           Em = value.start;
	   if (getNextValue(inputFile, &value))
	           Km = value.start;
	   if (getNextValue(inputFile, &value))
```

```
                rd = value.start;
            if (getNextValue(inputFile, &value))
                d = value;
            if (getNextValue(inputFile, &value))
                p = value;
        if (getNextValue(inputFile, &value))
                e0 = value.start;

            /* when finished, close the file */
            fclose(inputFile);
        }
        else        /* there was an error opening the file */
        {
            /* print out error to user that the file couldn't be opened */
            perror(fileName);
        }

        return returnValue;
}

int calculateVariables()
{
        /* place formulas for calculated variables here... */

        v = pow(rd / (rd + d.value), 3);
        R1 = 2 * (1 + 2 * p.value) * (1.0 - v);
        R2 = (1 + 2 * p.value) * (1 + 2 * v);
        R3 = 2*(1-p.value)*(2+v);
        R4 = 2*(1-p.value)*(1-v);
        R5 = (1-p.value)*(1+2*v);
        R6 = (2+p.value)*(2+v);
        R7 = (2+p.value)*(1-v);
        h  = Ep/Em; /*def*/
        R  = Em/Es.value;
        B  = R1+(h*R2+R3)*R+h*R4*pow(R,2);
        D  = R4+(h*R5+R6)*R+h*R7*pow(R,2);
        Eh = Em*B/D;
        G  = Kp/Km;
        S  = Km/Ks.value;
        A  = R1+(G*R2+R3)*S+G*R4*pow(S,2);
        C  = R4+(G*R5+R6)*S+G*R7*pow(S,2);
        E  = 2*R1+(G*R2+R3)*S+(h*R2+R3)*R+R4*(G+h)*S*R;
        F  = 2*R4+(G*R5+R6)*S+(h*R5+R6)*R+R7*(G+h)*S*R;
        U  = A/C;
        El = Em*(U-S*(U*F-E)/(R*C));
        T1 = (F+sqrt(pow(F,2)-4*C*D))/(2*C);
        T2 = (F-sqrt(pow(F,2)-4*C*D))/(2*C);
        Tp = e0*(Es.value/Ks.value)*T1;
        Tq = e0*(Es.value/Ks.value)*T2;
        Dli= Em*(1-S/R*T1)*(U*pow(T1,2)-E*T1/C+B/C)/((T1-T2)*T1);
        Dih= Em*(1-S/R*T2)*(U*pow(T2,2)-E*T2/C+B/C)/((T2-T1)*T2);
        Dlh= Dli+Dih;
        Dlhp=El-Eh;
        Kl = Km*U;
        Dil= Dli*e0/Tp;
        Dhi= Dih*e0/Tq;
        Dhl= Dil+Dhi;
        Kh = Km*(B/D-R*((B/D)*F-E)/(S*D));
```

```
	Dhlp=Kh-Kl;
	Q  = (Kh/Km-Eh/Em)*Tp*Tq*pow(Km/(e0*Em),2) +
				(Eh/Em-Kl/Km)*(Tp+Tq)*Km/(e0*Em) + Kl/Km - El/Em;
					return 1;
}

/* outputVariables() */

int outputVariables(FILE *output)
{
	fprintf(output, "Ep   = %.4e\n", Ep);
	fprintf(output, "Kp   = %.4e\n", Kp);
	fprintf(output, "Es   = %.4e (%s value)\n", Es.value,
			(Es.value == Es.start? "start": "incr"));
	fprintf(output, "Ks   = %.4e (%s value)\n", Ks.value,
			(Ks.value == Ks.start? "start": "incr"));
	fprintf(output, "Em   = %.4e\n", Em);
	fprintf(output, "Km   = %.4e\n", Km);
	fprintf(output, "rd   = %.4e\n", rd);
	fprintf(output, "d    = %.4e (%s value)\n", d.value,
			(d.value == d.start? "start": "incr"));
	fprintf(output, "p    = %.4e (%s value)\n", p.value,
			(p.value == p.start? "start": "incr"));
	fprintf(output, "\n");
	fprintf(output, "El   = %.4e\n", El);
	fprintf(output, "Eh   = %.4e\n", Eh);
	fprintf(output, "Tp   = %.4e\n", Tp);
	fprintf(output, "Tq   = %.4e\n", Tq);
	fprintf(output, "Dli  = %.4e\n", Dli);
	fprintf(output, "Dih  = %.4e\n", Dih);
	fprintf(output, "Kl   = %.4e\n", Kl);
	fprintf(output, "Kh   = %.4e\n", Kh);
	fprintf(output, "Dil  = %.4e\n", Dil);
	fprintf(output, "Dhi  = %.4e\n", Dhi);
	fprintf(output, "\n");
	fprintf(output, "Dlh  = %.4e\n", Dlh);
	fprintf(output, "Dlhp = %.4e\n", Dlhp);
	fprintf(output, "Dhl  = %.4e\n", Dhl);
	fprintf(output, "Dhlp = %.4e\n", Dhlp);
	fprintf(output, "Q    = %.4e\n", Q);
	fprintf(output, "----------------------------------------\n");
	return 1;
}

int variableNeedsIteration(struct inputVar *var)
{
	int needsIt;

	needsIt = 0;
	if (var && var->incr != 0.0)
	{
	   if (var->incr > 0.0)
			needsIt = (var->value + var->incr <= var->stop);
	   if (var->incr < 0.0)
			needsIt = (var->value + var->incr >= var->stop);

	   if (needsIt)
			var->value += var->incr;
```

```
	}
	return needsIt;
}

/* do not alter firstIteration line,
 * set it equal to one for the first iteration of
 * the input variables (this happens regardless of
 * whether or not any of the variables increments)
 */
int firstIteration = 1;

int haveAnotherIteration(FILE *output)
{
	int iterate;

	iterate = 0;
	if (firstIteration == 1)
	{
	   iterate = 1;
	   firstIteration = 0;
	}
	else if (output)
	{
	   /* see if any of the input variables need to be
	    * incremented
	    * this function does the necessary iteration
	    */
	   if (variableNeedsIteration(&Es))
	   {
		iterate = 1;
		printf("Es new value: %le\n", Es.value);
	   }
	   else if (variableNeedsIteration(&Ks))
	   {
		iterate = 1;
		printf("Ks new value: %le\n", Ks.value);
		Es.value = Es.start;
	   }
	   else if (variableNeedsIteration(&d))
	   {
		iterate = 1;
		printf("d new value: %le\n", d.value);
		Es.value = Es.start;
		Ks.value = Ks.start;
	   }
	   else if (variableNeedsIteration(&p))
	   {
		iterate = 1;
		printf("p new value: %le\n", p.value);
		Es.value = Es.start;
		Ks.value = Ks.start;
		d.value  = d.start;
	   }
	}


	return iterate;
}
```

```
/* main() */

int main(int argc, char *argv[])
{
	FILE *output;
	char *outFileName = "dielec.out";

	readUserVariables("dielec.in");

	output = fopen(outFileName, "w");
	if (output)
	{
	   fprintf(output, "; %s -- output file\n\n", outFileName);

	   /* loop as long as the input variables have additional
	    * iterations
	    */
	   while (haveAnotherIteration(output))
	   {
		calculateVariables();

		outputVariables(output);
	   }

	   /* when finished, close the file */
	   fclose(output);
	}
	else	/* error opening fileName */
	{
	   perror(outFileName);
	}
	return 0;
}
```

Output parameters

; dielec.out -- output file

Ep = 9.5700e+01
Kp = 4.6400e-02
Es = 2.6500e+00 (start value)
Ks = 1.0000e-20 (start value)
Em = 7.8600e+01
Km = 7.4700e-04
rd = 7.2425e+02
d = 4.6000e+00 (start value)
p = 2.0100e-01 (start value)

El = 2.1287e+02
Eh = 7.8372e+01
Tp = 2.7793e-06
Tq = 4.4739e-08
Dli = 1.0461e+02
Dih = 2.9894e+01
Kl = 5.4235e-04
Kh = 6.7889e-03

```
Dil  = 3.3310e-04
Dhi  = 5.9135e-03

Dlh  = 1.3450e+02
Dlhp = 1.3450e+02
Dhl  = 6.2466e-03
Dhlp = 6.2466e-03
Q    = -1.1744e-15
----------------------------------------
```

Program 2

Input parameters

```
; dielec2.in -- input file
;    order of variables is fixed in dielec2.c
;    do not change order.

; -- each variable in the form:
; start increment stop
; use increment 0.0 for no iteration

; Tp
        80.7e-9 0.0 80.66e-9
; g
        1 0.0 21.0
; h
        1.0 0.0 2.51
; El
        76.8 0.0 260.5
; Em
        71.4 0.0 85.86
; Eh
        70.6 0.0 84.6
; rd
        80 0.0 390.3
; d
        3.0 0.0 3.51
; p
        0.1 0.0 0.301
; u
        1.1 0.01 1.17
; e0
        8.85e-12

; output file name
dielec2.out
```

Program 2

```
/* dielec2.c */
/* Program to find phase parameters from dispersion data */

#include <stdio.h>
#include <math.h>
#include <string.h>

/* user variables */
```

```
/* double uses 8 bytes to store 1 number */

struct inputVar
{ double start, incr, stop, value;};

struct inputVar Tp;
struct inputVar g;
struct inputVar h;
struct inputVar El;
struct inputVar Em;
struct inputVar Eh;
struct inputVar rd;
struct inputVar d;
struct inputVar p;
struct inputVar u;
double e0;

/* calculated variables */

double v;
double R1;
double R2;
double R3;
double R4;
double R5;
double R6;
double R7;
double t;
double a1;
double b1;
double C1;
double D1;
double R;
double Es;
double B;
double D;
double RBD;
double U;
double a2;
double b2;
double C2;
double D2;
double S;
double A;
double C;
double E;
double F;
double RAC;
double Elp;
double T1;
double T2;
double REK;
double Tq;
double Ks;
double Km;
double Dli;
double Kl;
double Dil;
```

```
double Dhi;
double Dhl;
double Kh;
double Dhlp;
double Dih;
double Dlh;
double Dlhp;
double Q;

/* getNextValue() */

int getNextValue(FILE *file, struct inputVar *valuePtr)
{
      int success = 0;
      char line[295];

      while (fgets(line, 294, file) && !isdigit(line[0]));
      if (isdigit(line[0]))
      {
         sscanf(line, "%le %le %le",
                        &(valuePtr->start),
                        &(valuePtr->incr),
                        &(valuePtr->stop));
         valuePtr->value = valuePtr->start;
         success = 1;
      }
      return success;
}

/* readUserVariables() */

int readUserVariables(char *fileName)
{
      int returnValue = 0;
      FILE *inputFile;
      struct inputVar value;

      /* open the file */
      inputFile = fopen(fileName, "r");
      if (inputFile)
      {
         /* file opened successfully */

         /* change this section if you need a different
          set of input variables */

         if (getNextValue(inputFile, &value))
                  Tp = value;
         if (getNextValue(inputFile, &value))
                  g = value;
         if (getNextValue(inputFile, &value))
                  h = value;
         if (getNextValue(inputFile, &value))
                  El = value;
         if (getNextValue(inputFile, &value))
                  Em = value;
         if (getNextValue(inputFile, &value))
                  Eh = value;
```

```
        if (getNextValue(inputFile, &value))
                rd = value;
        if (getNextValue(inputFile, &value))
                d = value;
        if (getNextValue(inputFile, &value))
                p = value;
        if (getNextValue(inputFile, &value))
      u = value;
        if (getNextValue(inputFile, &value))
                e0 = value.start;

        /* when finished, close the file */
        fclose(inputFile);
    }
    else        /* there was an error opening the file */
    {
        /* print out error to user that the file couldn't be opened */
        perror(fileName);
    }

    return returnValue;
}

int calculateVariables()
{
    /* place formulas for calculated variables here... */

    v = pow(rd.value / (rd.value + d.value), 3);
    R1 = 2 * (1 + 2 * p.value) * (1.0 - v);
    R2 = (1 + 2 * p.value) * (1 + 2 * v);
    R3 = 2 * (1-p.value) * (2+v);
    R4 = 2 * (1-p.value) * (1-v);
    R5 = (1-p.value) * (1+2*v);
    R6 = (2+p.value) * (2+v);
    R7 = (2+p.value) * (1-v);
    t  = Eh.value / Em.value;
    a1 = h.value * (t*R7 - R4);
    b1 = (h.value * R2 + R3) - t * (h.value * R5 + R6);
    C1 = t*R4 - R1;
    D1 = sqrt (pow(b1,2) - 4*a1*C1);
    R  = (b1 + D1) / (2*a1);
    Es = Em.value / R;
    B  = R1 + (h.value*R2 + R3)*R + h.value*R4 * pow(R,2);
    D  = R4 + (h.value*R5 + R6)*R + h.value*R7 * pow(R,2);
    RBD= B / D;
        a2   = g.value * (u.value*R7 - R4);
   b2   = (g.value*R2 + R3) - u.value * (g.value*R5 + R6);
        C2   = u.value*R4 - R1;
        D2   = sqrt ( pow(b2,2) - 4*a2*C2 );
        S    = (b2 + D2) / (2 * a2);
        A    = R1 + (g.value*R2 + R3)*S + g.value*R4*pow(S,2);
        C    = R4 + (g.value*R5 + R6)*S + g.value*R7*pow(S,2);
        E    = 2*R1 + (g.value*R2 + R3)*S +
                        (h.value*R2 + R3)*R + R4*(g.value + h.value)*S*R;
        F    = 2*R4 + (g.value*R5 + R6)*S +
                        (h.value*R5 + R6)*R + R7*(g.value + h.value)*S*R;
    Elp = Em.value * (u.value - S*(u.value*F-E) / (R*C));
    T1 = (F + sqrt(pow(F,2) - 4*C*D)) / (2*C);
```

```
        T2 = (F - sqrt(pow(F,2) - 4*C*D)) / (2*C);
        REK = Tp.value / T1;
        Tq = REK * T2;
        Ks = (e0 * Es) / REK;
        Km = S * Ks;
        Dli = Em.value * (1 - S*T1/R) *
                (u.value*pow(T1,2) - E*T1/C + B/C) /
                ((T1 - T2)*T1);
        Dih = Em.value * (1 - S*T2/R) *
                (u.value*pow(T2,2) - E*T2/C + B/C) /
                ((T2 - T1)*T2);
        Dlh = Dli + Dih;
        Dlhp = Elp - Eh.value;
        Kl  = Km * u.value;
        Dil = Dli *e0/Tp.value;
        Dhi = Dih *e0/Tq;
        Dhl = Dil + Dhi;
        Kh  = Km * (B/D-R * ( (B/D) * F-E) / (S*D) );
        Dhlp= Kh - Kl;
        Q   = (Kh/Km - Eh.value/Em.value) *
                Tp.value * Tq * pow(Km/(e0*Em.value),2) +
                (Eh.value/Em.value - Kl/Km) * (Tp.value + Tq) *
                Km/(e0*Em.value) + Kl/Km - El.value/Em.value;

        return 1;
}

/* outputVariables() */

int outputVariables(FILE *output)
{
        fprintf(output, "Tp   = %.4e (%s value)\n", Tp.value,
                        (Tp.value == Tp.start? "start": "incr"));
        fprintf(output, "g    = %.4e (%s value)\n", g.value,
                        (g.value == g.start? "start": "incr"));
        fprintf(output, "h    = %.4e (%s value)\n", h.value,
                        (h.value == h.start? "start": "incr"));
        fprintf(output, "El   = %.4e (%s value)\n", El.value,
                        (El.value == El.start? "start": "incr"));
        fprintf(output, "Em   = %.4e (%s value)\n", Em.value,
                        (Em.value == Em.start? "start": "incr"));
        fprintf(output, "Eh   = %.4e (%s value)\n", Eh.value,
                        (Eh.value == Eh.start? "start": "incr"));
        fprintf(output, "rd   = %.4e (%s value)\n", rd.value,
                        (rd.value == rd.start? "start": "incr"));
        fprintf(output, "d    = %.4e (%s value)\n", d.value,
                        (d.value == d.start? "start": "incr"));
        fprintf(output, "p    = %.4e (%s value)\n", p.value,
                        (p.value == p.start? "start": "incr"));
           fprintf(output, "u    = %.4e (%s value)\n", u.value,
                                                                (u.value        ==
u.start? "start": "incr"));

                fprintf(output, "Elp  = %.4e\n", Elp);
                fprintf(output, "Es   = %.4e\n", Es);
                fprintf(output, "RBD  = %.4e\n", RBD);
                fprintf(output, "Km   = %.4e\n", Km);
                fprintf(output, "Ks   = %.4e\n", Ks);
```

```
                fprintf(output, "Tq   = %.4e\n", Tq);
                fprintf(output, "Dli  = %.4e\n", Dli);
                fprintf(output, "Dih  = %.4e\n", Dih);
                fprintf(output, "\n");
                fprintf(output, "R    = %.4e\n", R);
                fprintf(output, "S    = %.4e\n", S);
                fprintf(output, "Dlh  = %.4e\n", Dlh);
                fprintf(output, "Dlhp = %.4e\n", Dlhp);
                fprintf(output, "Kl   = %.4e\n", Kl);
                fprintf(output, "Dil  = %.4e\n", Dil);
                fprintf(output, "Dhi  = %.4e\n", Dhi);
                fprintf(output, "Dhl  = %.4e\n", Dhl);
                fprintf(output, "Kh   = %.4e\n", Kh);
                fprintf(output, "Dhlp = %.4e\n", Dhlp);
                fprintf(output, "Q    = %.4e\n", Q);

        fprintf(output, "\n");
        fprintf(output, "-----------------------------------------------\n");
        return 1;
}
int variableNeedsIteration(struct inputVar *var)
{
        int needsIt;

        needsIt = 0;
        if (var && var->incr != 0.0)
        {
            if (var->incr > 0.0)
                    needsIt = (var->value + var->incr <= var->stop);
            if (var->incr < 0.0)
                    needsIt = (var->value + var->incr >= var->stop);

            if (needsIt)
                    var->value += var->incr;
        }
        return needsIt;
}

/* do not alter firstIteration line,
 * set it equal to one for the first iteration of
 * the input variables (this happens regardless of
 * whether or not any of the variables increments)
 */
int firstIteration = 1;

int haveAnotherIteration(FILE *output)
{
        int iterate;

        iterate = 0;
        if (firstIteration == 1)
        {
            iterate = 1;
            firstIteration = 0;
        }
        else if (output)
        {
            /* see if any of the input variables need to be
```

```
 * incremented
 * this function does the necessary iteration
 */
if (variableNeedsIteration(&Tp))
{
	iterate = 1;
	printf("Tp new value: %le\n", Tp.value);
}
else if (variableNeedsIteration(&g))
{
	iterate = 1;
	printf("g new value: %le\n", g.value);
	Tp.value = Tp.start;
}
else if (variableNeedsIteration(&h))
{
	iterate = 1;
	printf("h new value: %le\n", h.value);
	Tp.value = Tp.start;
	g.value = g.start;
}
else if (variableNeedsIteration(&El))
{
	iterate = 1;
	printf("El new value: %le\n", El.value);
	Tp.value = Tp.start;
	g.value = g.start;
	h.value = h.start;
}
else if (variableNeedsIteration(&Em))
{
	iterate = 1;
	printf("Em new value: %le\n", Em.value);
	Tp.value = Tp.start;
	g.value = g.start;
	h.value = h.start;
	El.value = El.start;
}
else if (variableNeedsIteration(&Eh))
{
	iterate = 1;
	printf("Eh new value: %le\n", Eh.value);
	Tp.value = Tp.start;
	g.value = g.start;
	h.value = h.start;
	El.value = El.start;
	Em.value = Em.start;
}
else if (variableNeedsIteration(&rd))
{
	iterate = 1;
	printf("rd new value: %le\n", rd.value);
	Tp.value = Tp.start;
	g.value = g.start;
	h.value = h.start;
	El.value = El.start;
	Em.value = Em.start;
	Eh.value = Eh.start;
```

```
        }
        else if (variableNeedsIteration(&d))
        {
                iterate = 1;
                printf("d new value: %le\n", d.value);
                Tp.value = Tp.start;
                g.value = g.start;
                h.value = h.start;
                El.value = El.start;
                Em.value = Em.start;
                Eh.value = Eh.start;
                rd.value = rd.start;
        }

        else if (variableNeedsIteration(&p))
        {
                iterate = 1;
                printf("p new value: %le\n", p.value);
                Tp.value = Tp.start;
                g.value = g.start;
                h.value = h.start;
                El.value = El.start;
                Em.value = Em.start;
                Eh.value = Eh.start;
                rd.value = rd.start;
                d.value = d.start;
        }
           else if (variableNeedsIteration(&u))
        {
                iterate = 1;
                printf("u new value: %le\n", u.value);
                Tp.value = Tp.start;
                g.value = g.start;
                h.value = h.start;
                El.value = El.start;
                Em.value = Em.start;
                Eh.value = Eh.start;
                rd.value = rd.start;
                d.value = d.start;
                p.value = p.start;
        }
      }

        return iterate;
}

/* main() */

int main(int argc, char *argv[])
{
      FILE *output;
      char *outFileName = "dielec2.out";

      readUserVariables("dielec2.in");

      output = fopen(outFileName, "w");
      if (output)
      {
```

```
        fprintf(output, "; %s -- output file\n\n", outFileName);

        while (haveAnotherIteration(output))
        {
                        calculateVariables();

                        outputVariables(output);
        }
        fclose(output);
    }
    else        /* error opening output file */
    {
        perror(outFileName);
    }

    return 0;
}
```

Output parameters

; dielec2.out -- output file

Tp = 8.0700e-08 (start value)
g   = 1.0000e+00 (start value)
h   = 1.0000e+00 (start value)
El = 7.6800e+01 (start value)
Em = 7.1400e+01 (start value)
Eh = 7.0600e+01 (start value)
rd = 8.0000e+01 (start value)
d   = 3.0000e+00 (start value)
p   = 1.0000e-01 (start value)
u   = 1.1600e+00 (incr value)
Elp = 7.6852e+01
Es = 2.3845e+01
RBD = 9.8880e-01
Km = 4.0538e-03
Ks = 1.7492e-01
Tq = 1.2981e-09
Dli = 5.5576e+00
Dih = 6.9459e-01

R   = 2.9943e+00
S   = 2.3175e-02
Dlh = 6.2522e+00
Dlhp = 6.2522e+00
Kl = 4.7024e-03
Dil = 6.0948e-04
Dhi = 4.7356e-03
Dhl = 5.3451e-03
Kh = 1.0047e-02
Dhlp = 5.3451e-03
Q   = 7.3168e-04