

```

1  #include <iostream>
2  #include <math.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <windows.h>
6  #include <complex>
7
8
9  using namespace std;
10
11 double** alloc_matrix(long row, long column);
12 void free_matrix(double** mtrx, long row);
13 void free_boolmatrix(int** mtrx, long row);
14 int** alloc_boolmatrix(long row, long column);
15 int* alloc_intvector(long size);
16 double* alloc_vector(long size);
17 bool** alloc_boolboolmatrix(long row, long column);
18 complex <double>* alloc_cvector(long size);
19
20 int IrrRefBottom(double* R_Bot_2d_1, double* G_Bot_2d_1, double* B_Bot_2d_1, double*
    Params, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int
    MaxY, double* bw, double* Absorbance, double* f_fine, double MieC);
21 double CalcFuresnel(double theta_In, double theta_Out);
22 void Solve4thEq_DLL(bool& Hitpath111, double& b11x, double& b11y, double& b11z,
    double x11, double y11, double z11, double x41, double y41, double z41, double
    n12);
23 int DKA_rh(double* Coeff, double* zr, double* zi, int n_dim, double deltax);
24 int DenstFlucSca(double* R_ES_2d_1, double* G_ES_2d_1, double* B_ES_2d_1, double*
    Params1, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int
    MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double MieC);
25 int MieSca(double* R_Mie_2d_1, double* G_Mie_2d_1, double* B_Mie_2d_1, double*
    Params1, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int
    MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double MieC);
26 int splineDLL64(int ScreenX, int ScreenY, double* nn44_1, int* nn55_1, double
    smax_thre, long MaxIter);
27 int PreSplineLake64(int* T88_1, double** LinHomG, double** LinIHomG, int
    ScreenTrans, int ScreenLong, int ScreenOrigTrans, int ScreenOrigLong, int NumStep,
    double epsilon, double r1, double FacDetailed, double** R_x_2d, double** G_x_2d,
    double** B_x_2d, double* R88_1, double* G88_1, double* B88_1, int*
    ScreenFullFlag_1, double* MaxYY, double& MinX1, double& MaxX1, double& MinY1,
    double& MaxY1);
28 void XY2XYXY(double epsilon, double x0, double& x11, double y0, double& y1);
29 int ConvertPhoto2Circle2(int ScreenTrans, int ScreenLong, int ScreenOrigTrans, int
    ScreenOrigLong, bool FlagInv, double& XXX, double& ZZZ, double** LinHomG, double**
    LinIHomG);
30 double Max2(double x, double y);
31 void BHMIE1_array(int n1, double deltaLamda, double* s11, int nang, double refmed,
    double refre, double refim, double radius_1, double wavel, double& QSCA_ret);
32 double BHMIE1_fromTable(double Angle, double* s11, int nang);
33 int DenstFlucSca_Rect_2nd(double* Iz1d, double* R_ES_2d_1, double* G_ES_2d_1,
    double* B_ES_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int
    MaxX, int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double
    MieC, double* MieScatC_array);
34 int IrrRefBottom_Rect_2nd(double* Iz1d, double* R_Bot_2d_1, double* G_Bot_2d_1,
    double* B_Bot_2d_1, double* Params, double* Xbar, double* Ybar, double* Zbar, int
    MaxX, int MaxY, double* bw, double* Absorbance, double* f_fine, double MieC,
    double* MieScatC_array);
35 int MieSca_Rect_2nd(double* Iz1d, double* R_Mie_2d_1, double* G_Mie_2d_1, double*
    B_Mie_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int MaxX,
    int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double MieC,
    double* MieScatC_array);
36 int MieScat_Sca_Rect_2nd(double* Iz1d, int nang, double** s11_table, double refmed,
    double refre, double refim, double radius, double* R_Mie_2d_1, double* G_Mie_2d_1,
    double* B_Mie_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int

```

```

MaxX, int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double
Miec, double* MieScatC_array);
37 int MieScat_Give2(int nang, double** s11_array, double& a, double& b, int n1, double
deltaLamda, double refmed, double refre, double refim, double radius_1, double*
MieScatC_array1);
38
39
40
41
42
43
44
45 double Max2(double x, double y)
46 {
47     if (x > y) return x;
48     else return y;
49 }
50 }
51
52 void BHMIE1_array(int n1, double deltaLamda, double* s11, int nang, double refmed,
double refre, double refim, double radius_1, double wavel, double& QSCA_ret)
53 {
54     complex <double> refrel;
55     refrel = complex <double>(refre, refim) / refmed;
56     double x;
57     x = 2.0 * 3.14159265358979 * radius_1 * refmed / wavel;
58
59     double alfa;
60     alfa = 3.14159265358979 * 2.0 * radius_1 / wavel;
61
62     double* amu;
63     double* theta;
64     double* PI;
65     double* TAU;
66     double* PIO;
67     double* PI1;
68
69     complex<double>* s1A;
70     complex <double>* s2A;
71
72     amu = alloc_vector(nang + 10);
73     theta = alloc_vector(nang + 10);
74     PI = alloc_vector(nang + 10);
75     TAU = alloc_vector(nang + 10);
76     PIO = alloc_vector(nang + 10);
77     PI1 = alloc_vector(nang + 10);
78
79     s1A = alloc_cvector(2 * nang + 10);
80     s2A = alloc_cvector(2 * nang + 10);
81
82
83     complex <double> d[3000];
84     complex <double> y, xi, xi0, xi1, AN, BN;
85     double PSIO, PSI1, CHIO, CHI1, PSI, APSIO, APSI1, dn, dx;
86     double xstop;
87     int nstop;
88     double ymod;
89     int nmx;
90     double dang;
91     int j;
92
93     double fn;
94     double APSI;
95     double CHI;

```

```

96
97     double QSCA;
98
99
100    dx = x;
101    y = x * refrel;
102    xstop = x + 4.0 * pow(x, 0.3333) + 2.0;
103    nstop = static_cast<int>(xstop);
104    ymod = abs(y);
105    nmx = static_cast<int>(Max2(xstop, ymod)) + 15;
106    dang = 3.14159265358979 / 2.0 / (static_cast<double>(nang - 1));
107
108
109    int nn;
110    int n;
111    double rn;
112
113    for (j = 1; j <= nang; j++)
114    {
115        theta[j] = (static_cast<double>(j - 1)) * dang;
116        amu[j] = cos(theta[j]);
117    }
118
119    d[nmx] = complex<double>(0.0, 0.0);
120    nn = nmx - 1;
121    for (n = 1; n <= nn; n++)
122    {
123        rn = static_cast<double>(nmx - n + 1);
124        d[nmx - n] = (rn / y) - (1.0 / (d[nmx - n + 1] + rn / y));
125    }
126
127    for (j = 1; j <= nang; j++)
128    {
129        PI0[j] = 0.0;
130        PI1[j] = 1.0;
131    }
132
133
134    nn = 2 * nang - 1;
135
136    for (j = 1; j <= nn; j++)
137    {
138        s1A[j] = complex<double>(0.0, 0.0);
139        s2A[j] = complex<double>(0.0, 0.0);
140    }
141
142    PSIO = cos(dx);
143    PSI1 = sin(dx);
144    CHIO = -sin(x);
145    CHI1 = cos(x);
146    APSIO = PSIO;
147    APSI1 = PSI1;
148    xi0 = complex<double>(APSIO, -CHI0);
149    xi1 = complex<double>(APSI1, -CHI1);
150    QSCA = 0.0;
151    n = 1;
152
153    do
154    {
155
156        dn = static_cast<double>(n);
157        rn = static_cast<double>(n);
158
159        fn = (2.0 * rn + 1.0) / (rn * (rn + 1.0));

```

```

160     PSI = (2.0 * dn - 1.0) * PSI1 / dx - PSIO;
161     APSI = PSI;
162     CHI = (2.0 * rn - 1.0) * CHI1 / x - CHIO;
163     xi = complex<double>(APSI, -CHI);
164
165     AN = (d[n] / refrel + rn / x) * APSI - APSI1;
166     AN = AN / ((d[n] / refrel + rn / x) * xi - xi1);
167     BN = (refrel * d[n] + rn / x) * APSI - APSI1;
168     BN = BN / ((refrel * d[n] + rn / x) * xi - xi1);
169
170     QSCA = QSCA + (2.0 * rn + 1.0) * (abs(AN) * abs(AN) + abs(BN) * abs(BN));
171
172     int jj;
173     double p;
174     double t;
175
176     for (j = 1; j <= nang; j++)
177     {
178         jj = 2 * nang - j;
179         PI[j] = PI1[j];
180         TAU[j] = rn * amu[j] * PI[j] - (rn + 1.0) * PIO[j];
181         p = pow(-1.0, n - 1);
182         s1A[j] = s1A[j] + fn * (AN * PI[j] + BN * TAU[j]);
183         t = pow(-1.0, n);
184         s2A[j] = s2A[j] + fn * (AN * TAU[j] + BN * PI[j]);
185         if (j != jj)
186         {
187             s1A[jj] = s1A[jj] + fn * (AN * PI[j] * p + BN * TAU[j] * t);
188             s2A[jj] = s2A[jj] + fn * (AN * TAU[j] * t + BN * PI[j] * p);
189         }
190     }
191
192     PSIO = PSI1;
193     PSI1 = PSI;
194     APSI1 = APSI;
195     CHIO = CHI1;
196     CHI1 = CHI;
197     xi1 = complex<double>(APSI1, -CHI1);
198     n = n + 1;
199     rn = static_cast<double>(n);
200
201     for (j = 1; j <= nang; j++)
202     {
203         PI1[j] = ((2.0 * rn - 1.0) / (rn - 1.0)) * amu[j] * PI[j];
204         PI1[j] = PI1[j] - rn * PIO[j] / (rn - 1.0);
205         PIO[j] = PI[j];
206     }
207 } while (n - 1 - nstop < 0);
208
209 QSCA = (2.0 / (x * x)) * QSCA;
210
211 double s11_val;
212
213 for (j = 1; j <= 2 * nang - 1; j++)
214 {
215     s11_val = 0.5 * abs(s2A[j]) * abs(s2A[j]);
216     s11_val = s11_val + 0.5 * abs(s1A[j]) * abs(s1A[j]);
217     s11[j] = s11_val;
218 }
219
220 QSCA_ret = QSCA;
221 }
222
223 int MieScat_Give2(int nang, double** s11_array, double& a, double& b, int n1, double >

```

```

    deltaLamda, double refmed, double refre, double refim, double radius_1, double*
    MieScatC_array1)
224 {
225     double Pi;
226     Pi = 3.14159265358979;
227
228     double wavel;
229     int i, j;
230
231     double X[100];
232     double Y[100];
233
234     for (i = 0; i < 100; i++)
235     {
236         X[i] = 0.0;
237         Y[i] = 0.0;
238     }
239
240     double SumX, SumY, SumXY, SumX2;
241
242     double QSCA_ret;
243
244     double* MieScatC_array;
245     MieScatC_array = alloc_vector(75);
246
247     double* s11;
248     s11 = alloc_vector(2 * nang+1);
249
250     for (i = 1; i <= n1; i++)
251     {
252         wavel = 380.0 + (static_cast<double>(i - 1)) * deltaLamda;
253         BHMIE1_array(n1, deltaLamda, s11, nang, refmed, refre, refim, radius_1,
            wavel / 1000.0, QSCA_ret);
254         for (j = 1; j <= 2 * nang - 1; j++)
255         {
256             s11_array[i][j] = s11[j];
257         }
258
259         MieScatC_array[i-1] = QSCA_ret;
260         X[i-1] = wavel;
261         Y[i-1] = MieScatC_array[i-1];
262     }
263
264     SumX = 0.0;
265     SumY = 0.0;
266     SumXY = 0.0;
267     SumX2 = 0.0;
268
269     for (i = 1; i < n1; i++)
270     {
271         SumX = SumX + log(X[i - 1]);
272         SumY = SumY + log(Y[i - 1]);
273         SumXY = SumXY + log(X[i - 1]) * log(Y[i - 1]);
274         SumX2 = SumX2 + log(X[i - 1]) * log(X[i - 1]);
275     }
276
277     double A, B;
278
279     double rnm1;
280
281     rnm1 = static_cast<double>(n1 - 1);
282
283
284     B = (-SumX * SumY + rnm1 * SumXY) / (rnm1 * SumX2 - SumX * SumX);

```

```

285     A = 1.0 / rnm1 * (SumY - B * SumX);
286
287     a = exp(A);
288     b = B;
289
290     SumY = 0.0;
291
292     if (b >= 0.0)
293     {
294         for (i = 1; i < n1; i++)
295         {
296             SumY = SumY + MieScatC_array[i-1];
297         }
298         SumY = SumY / (static_cast<double>(n1 - 1));
299         a = SumY;
300         b = 0;
301         for (i = 1; i <= n1; i++)
302         {
303             MieScatC_array[i-1] = SumY;
304         }
305     }
306     else
307     {
308         for (i = 1; i <= n1; i++)
309         {
310             MieScatC_array[i-1] = a * pow(X[i-1], b);
311         }
312     }
313
314     for (i = 1; i <= n1; i++)
315     {
316         MieScatC_array1[i-1] = MieScatC_array[i-1];
317     }
318     return 1;
319 }
320 }
321
322 int Making_Table_SolarColor(double* Xbar, double* Ybar, double* Zbar, double* AMO,
double k_aero, double theta8, double fai8, double currtheta, double currfai,
double& R3, double& G3, double& B3)
323 {
324     double n1;
325     double H_h;
326     double r2;
327     double ramda;
328     double ganma2;
329
330     int iz_incre;
331     int k;
332     double wl;
333
334     double* IntegralExp;
335     IntegralExp = alloc_vector(80);
336
337     double CurrR;
338     double CurrG;
339     double CurrB;
340
341     double CurrX;
342     double CurrY;
343     double CurrZ;
344
345     double Pi;
346

```

```

347     Pi = 3.14159265358979;
348     n1 = 1.0002;
349
350     H_h = 7994.0;
351
352     double DistDelta;
353     DistDelta = 100.0;
354
355     double* A1;
356     A1 = alloc_vector(5);
357
358     double* B1;
359     B1 = alloc_vector(5);
360
361     A1[0] = -sin(theta8) * cos(fai8);
362     A1[1] = -sin(theta8) * sin(fai8);
363     A1[2] = -cos(theta8);
364
365     double Km;
366     double CosGamma;
367
368     double dr;
369     double kx;
370     double phasem;
371     double ka;
372     double phasea;
373
374     double term1;
375     double term2;
376     double term3;
377     double term4;
378
379     B1[0] = -sin(currtheta) * cos(currfai);
380     B1[1] = -sin(currtheta) * sin(currfai);
381     B1[2] = -cos(currtheta);
382
383     CosGamma = A1[0] * B1[0] + A1[1] * B1[1] + A1[2] * B1[2];
384     gamma2 = acos(CosGamma);
385
386     for (k = 1; k <= 70; k++)
387     {
388         IntegralExp[k] = 0.0;
389         wl = 380.0 + (static_cast<double>(k - 1)) * 5.0;
390         ramda = wl * 1.0e-9;
391
392         iz_incre = 1;
393         do
394         {
395             r2 = static_cast<double>(iz_incre) * DistDelta / cos(currtheta);
396             dr = DistDelta / cos(currtheta);
397
398             kx = 8.0 * Pi * Pi * Pi * (n1 * n1 - 1.0) * (n1 * n1 - 1.0) / (3.0 *
399                 ramda * ramda * ramda * ramda * 2.5473E+25);
400             phasem = 0.7629 * (1.0 + 0.932 * CosGamma * CosGamma) / (4.0 * Pi);
401
402             ka = k_aero * 550.0 / wl;
403
404             phasea = 0.02 * (1.0 + 9.0 * pow(cos(gamma2 / 2.0), 16.0));
405
406             term1 = log(kx * phasem + ka * phasea);
407             term2 = -(kx + ka) * (r2 + (H_h - r2 * cos(currtheta)) / cos(theta8));
408             term3 = 0.0;
409             term4 = log(dr);

```

```

410
411         IntegralExp[k] = IntegralExp[k] + exp(term1 + term2 + term3 + term4);
412
413         iz_incre = iz_incre + 1;
414         if (r2 > H_h / cos(currtheta)) goto P226;
415     } while (1);
416 P226:
417     ;
418 }
419
420
421 CurrX = 0.0;
422 CurrY = 0.0;
423 CurrZ = 0.0;
424
425 Km = 683.0;
426
427 for (k = 1; k <= 70; k++)
428 {
429     CurrX = CurrX + IntegralExp[k] * Xbar[k] * Km * AMO[k] * 5.0;
430     CurrY = CurrY + IntegralExp[k] * Ybar[k] * Km * AMO[k] * 5.0;
431     CurrZ = CurrZ + IntegralExp[k] * Zbar[k] * Km * AMO[k] * 5.0;
432 }
433
434 CurrR = 3.240625477 * CurrX - 1.537207972 * CurrY - 0.498628599 * CurrZ;
435 CurrG = -0.968930715 * CurrX + 1.875756061 * CurrY + 0.041517524 * CurrZ;
436 CurrB = 0.05571012 * CurrX - 0.204021051 * CurrY + 1.056995942 * CurrZ;
437
438 if (CurrR < 0.0) CurrR = 0.0;
439 if (CurrG < 0.0) CurrG = 0.0;
440 if (CurrB < 0.0) CurrB = 0.0;
441
442 R3 = CurrR;
443 G3 = CurrG;
444 B3 = CurrB;
445
446 return 1;
447 }
448
449
450 int main(int argc, char* argv[])
451 {
452     double FactorDC;
453     double gzai;
454     double Sigma;
455     double delta;
456     double deltaLamda;
457     double r1;
458     double d;
459     double Xgrid;
460     double Ygrid;
461     double FacDetailed;
462     double LakeShoreEPS;
463     double fai77;
464     double theta77;
465     double n12;
466     double xobs;
467     double yobs;
468     double zobs;
469     double k1;
470     double Km;
471     double SettleShodo;
472     double TheoShodo;
473     double Rittaikaku;

```



```
474     double refl;
475     double iLambertian;
476     double delta0;
477     double FactorZ;
478     double FlagStepAgainstSlope;
479     double InitZ;
480     double theta1;
481     double y0;
482
483     //-----
484     int FlagCliff;
485     double* Xbar;
486     Xbar = alloc_vector(80);
487     double* Ybar;
488     Ybar = alloc_vector(80);
489     double* Zbar;
490     Zbar = alloc_vector(80);
491     double* MieTheoScat;
492     MieTheoScat = alloc_vector(80);
493     double* AMO;
494     AMO = alloc_vector(80);
495
496     int MaxX, MaxY;
497     int Maxk;
498
499     double* bw;
500     bw = alloc_vector(80);
501     double* Absorbance;
502     Absorbance = alloc_vector(80);
503     double* f_fine;
504     f_fine = alloc_vector(80);
505     double MieC;
506
507     double Miea0;
508     double Miea1;
509     double Miea2;
510     double Miea3;
511     double Miea4;
512
513     double iPetzoldPF;
514
515     int i, j;
516
517     int Ndim;
518
519     double* param3;
520     param3 = alloc_vector(40);
521     for (i = 0; i < 40; i++)
522         param3[i] = 0.0;
523
524     double dummy;
525
526     double** HomG;
527     HomG = alloc_matrix(4, 4);
528
529     double** iHomG;
530     iHomG = alloc_matrix(4, 4);
531
532     int ScreenTrans;
533     int ScreenLong;
534     int ScreenOrigTrans;
535     int ScreenOrigLong;
536     double epsilon;
537
```

```

538     char* TypeCalc;
539     TypeCalc = argv[1];
540
541     if (TypeCalc[0] == 'T')
542     {
543         double k_aero;
544         double theta8;
545         double fai8;
546         double currtheta;
547         double currfai;
548         double** R3;
549         double** G3;
550         double** B3;
551
552         double dummy5;
553
554         double Pi;
555         Pi = 3.14159265358979;
556
557         int NumIncreTheta2;
558         int NumIncreFai2;
559
560         int responce;
561
562         double R33, G33, B33;
563
564         FILE* InputFileName;
565         if ((fopen_s(&InputFileName, "Set_Making_Table_SolarColor.txt", "r")) != 0)
566             return 0;
567         for (i = 1; i <= 70; i++)
568         {
569             fscanf_s(InputFileName, "%lf", &dummy5);
570             Xbar[i] = dummy5;
571         }
572         for (i = 1; i <= 70; i++)
573         {
574             fscanf_s(InputFileName, "%lf", &dummy5);
575             Ybar[i] = dummy5;
576         }
577         for (i = 1; i <= 70; i++)
578         {
579             fscanf_s(InputFileName, "%lf", &dummy5);
580             Zbar[i] = dummy5;
581         }
582         for (i = 1; i <= 70; i++)
583         {
584             fscanf_s(InputFileName, "%lf", &dummy5);
585             AMO[i] = dummy5;
586         }
587         fscanf_s(InputFileName, "%lf", &k_aero);
588         fscanf_s(InputFileName, "%lf", &theta8);
589         fscanf_s(InputFileName, "%lf", &fai8);
590         fscanf_s(InputFileName, "%d", &NumIncreTheta2);
591         fscanf_s(InputFileName, "%d", &NumIncreFai2);
592
593         fclose(InputFileName);
594
595         int i, j;
596
597         R3 = alloc_matrix(NumIncreTheta2, NumIncreFai2);
598         G3 = alloc_matrix(NumIncreTheta2, NumIncreFai2);
599         B3 = alloc_matrix(NumIncreTheta2, NumIncreFai2);
600
601         for (i = 0; i < NumIncreTheta2; i++)

```

```

601     for (j = 0; j < NumIncreFai2; j++)
602     {
603         currtheta = static_cast<double>(i) / static_cast<double>
        (NumIncreTheta2) * Pi / 2.0;
604         currfai = static_cast<double>(j) / static_cast<double>(NumIncreFai2)
        * 2.0 * Pi;
605         if (fabs(currtheta - 0.01) < 0.01)
606         {
607             currtheta = 1.0 / static_cast<double>(NumIncreTheta2) * 0.1;
608         }
609
610         responce = Making_Table_SolarColor(Xbar, Ybar, Zbar, AM0, k_aero,
        Pi / 2.0 - theta8, Pi - fai8, currtheta, currfai, R33, G33, B33);
611
612         R3[i][j] = R33;
613         G3[i][j] = G33;
614         B3[i][j] = B33;
615     }
616
617     FILE* DestMatrixFile;
618     int ret;
619     errno_t err = fopen_s(&DestMatrixFile,
        "Dest_Making_Table_SolarColor.txt", "w");
620
621     if (err != 0)
622     {
623         return 0;
624     }
625     else
626     {
627         ret = fprintf_s(DestMatrixFile, "%d¥n", NumIncreTheta2);
628         ret = fprintf_s(DestMatrixFile, "%d¥n", NumIncreFai2);
629         for (i = 0; i < NumIncreTheta2; i++)
630             for (j = 0; j < NumIncreFai2; j++)
631                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", R3[i][j]);
632
633         for (i = 0; i < NumIncreTheta2; i++)
634             for (j = 0; j < NumIncreFai2; j++)
635                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", G3[i][j]);
636
637         for (i = 0; i < NumIncreTheta2; i++)
638             for (j = 0; j < NumIncreFai2; j++)
639                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", B3[i][j]);
640
641         fclose(DestMatrixFile);
642         return 1;
643     }
644 }
645
646
647
648 if (TypeCalc[0] == 'H')
649 {
650     int nang;
651     int nwl;
652     double refmed;
653     double refre;
654     double refim;
655     double radius_1;
656
657     double a, b;
658
659
660     FILE* InputFileName;

```

```

661     if ((fopen_s(&InputFileName, "SetMieScat_para.txt", "r")) != 0)
662     {
663         std::cout << "Error in opening SetMieScat_para Press any key to
        continue...";
664         std::cin.get();
665         return 0;
666     }
667
668     fscanf_s(InputFileName, "%d", &nang);
669     fscanf_s(InputFileName, "%d", &nwl);
670     fscanf_s(InputFileName, "%lf", &deltaLamda);
671     fscanf_s(InputFileName, "%lf", &refmed);
672     fscanf_s(InputFileName, "%lf", &refre);
673     fscanf_s(InputFileName, "%lf", &refim);
674     fscanf_s(InputFileName, "%lf", &radius_1); // in unit of maicro
        meter'
675     fclose(InputFileName);
676
677     double** s11_array;
678     s11_array = alloc_matrix(71, 2 * nang);
679
680     for (i = 0; i <= 70; i++)
681         for (j=0; j<=2 * nang-1; j++)
682             s11_array[i][j] = 0.0;
683
684     double* MieScatC_array_1;
685     MieScatC_array_1 = alloc_vector(100);
686     for (i = 0; i < 100; i++)
687         MieScatC_array_1[i] = 0.0;
688
689
690     if (0 == MieScat_Give2(nang, s11_array, a, b, 70, 5.0, refmed, refre,
        refim, radius_1, MieScatC_array_1))
691     {
692         std::cout << "Error in MieScat_Give2 Press any key to continue...";
693         std::cin.get();
694         return 0;
695     }
696     else
697     {
698         FILE* DestMatrixFile;
699         int ret;
700         errno_t err = fopen_s(&DestMatrixFile, "SetMieScat_destmatrix.txt",
        "w");
701
702         if (err != 0)
703         {
704             return 0;
705         }
706         else
707         {
708             Ndim = 2 * nang;
709             ret = fprintf_s(DestMatrixFile, "%d¥n", Ndim);
710             for (i = 1; i <= 70; i++)
711                 for (j = 1; j <= 2 * nang-1 ;j++)
712                 {
713                     ret = fprintf_s(DestMatrixFile, "%8.10le¥n", s11_array[i]
                        [j]);
714                 }
715             Ndim = 70;
716             ret = fprintf_s(DestMatrixFile, "%d¥n", Ndim);
717             for (i = 1; i <= Ndim; i++)
718             {
719                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", MieScatC_array_1

```

```

        [i]);
720     }
721     ret = fprintf_s(DestMatrixFile, "%8.10le\n", a);
722     ret = fprintf_s(DestMatrixFile, "%8.10le\n", b);
723     fclose(DestMatrixFile);
724     return 1;
725 }
726 }
727 }
728
729 if (TypeCalc[0] == 'U')
730 {
731     int NwINBot;
732     int nang;
733     int Ncounts11;
734     double** s11_array;
735     double refmed, refre, refim, radius;
736
737     FILE* InputFileName;
738     if ((fopen_s(&InputFileName, "MieScat_para.txt", "r")) != 0) return 0;
739     for (i = 0; i <= 28; i++)
740     {
741         fscanf_s(InputFileName, "%lf", &dummy);
742         param3[i] = dummy;
743     }
744     fscanf_s(InputFileName, "%d", &FlagCliff);
745     for (i = 1; i <= 70; i++)
746     {
747         fscanf_s(InputFileName, "%lf", &dummy);
748         Xbar[i] = dummy;
749     }
750     for (i = 1; i <= 70; i++)
751     {
752         fscanf_s(InputFileName, "%lf", &dummy);
753         Ybar[i] = dummy;
754     }
755     for (i = 1; i <= 70; i++)
756     {
757         fscanf_s(InputFileName, "%lf", &dummy);
758         Zbar[i] = dummy;
759     }
760     fscanf_s(InputFileName, "%d", &MaxX);
761     fscanf_s(InputFileName, "%d", &MaxY);
762     fscanf_s(InputFileName, "%d", &Maxk);
763     for (i = 1; i <= 70; i++)
764     {
765         fscanf_s(InputFileName, "%lf", &dummy);
766         bw[i] = dummy;
767     }
768     for (i = 1; i <= 70; i++)
769     {
770         fscanf_s(InputFileName, "%lf", &dummy);
771         Absorbance[i] = dummy;
772     }
773     for (i = 1; i <= 70; i++)
774     {
775         fscanf_s(InputFileName, "%lf", &dummy);
776         f_fine[i] = dummy;
777     }
778     fscanf_s(InputFileName, "%lf", &Miec);
779     for (i = 1; i <= 70; i++)
780     {
781         fscanf_s(InputFileName, "%lf", &dummy);
782

```

```

783     MieTheoScat[i] = dummy;
784 }
785 fscanf_s(InputFileName, "%d", &NwINBot);
786
787 double* lz1d;
788 lz1d = alloc_vector(NwINBot);
789
790 for (i = 0; i < NwINBot; i++)
791 {
792     fscanf_s(InputFileName, "%lf", &dummy);
793     lz1d[i] = dummy;
794 }
795 fscanf_s(InputFileName, "%d", &nang);
796 fscanf_s(InputFileName, "%d", &Ncounts11);
797
798 s11_array = alloc_matrix(71, 2*nang);
799
800 for (i = 1; i <=70; i++)
801     for (j=1; j<=2*nang-1; j++)
802     {
803         fscanf_s(InputFileName, "%lf", &dummy);
804         s11_array[i][j] = dummy;
805     }
806 fscanf_s(InputFileName, "%lf", &refmed);
807 fscanf_s(InputFileName, "%lf", &refre);
808 fscanf_s(InputFileName, "%lf", &refim);
809 fscanf_s(InputFileName, "%lf", &radius);
810
811 fclose(InputFileName);
812
813 FactorDC = param3[0];
814 gzai = param3[1];
815 Sigma = param3[2];
816 delta = param3[3];
817 deltaLamda = param3[4];
818 r1 = param3[5];
819 d = param3[6];
820 Xgrid = param3[7];
821 Ygrid = param3[8];
822 FacDetailed = param3[9];
823 LakeShoreEPS = param3[10];
824 fai77 = param3[11];
825 theta77 = param3[12];
826 n12 = param3[13];
827 xobs = param3[14];
828 yobs = param3[15];
829 zobs = param3[16];
830 k1 = param3[17];
831 Km = param3[18];
832 SettleShodo = param3[19];
833 TheoShodo = param3[20];
834 Rittaikaku = param3[21];
835 refl = param3[22];
836 delta0 = param3[23];
837 FactorZ = param3[24];
838 FlagStepAgainstSlope = param3[25];
839 InitZ = param3[26];
840 theta1 = param3[27];
841 iPetzoldPF = param3[28];
842
843 Ndim = (MaxX * static_cast<int>(FacDetailed) + 1) * (MaxY * static_cast<int> ➤
      (FacDetailed) + 1);
844
845 double* R_MieScat_2d_1;

```

```

846 R_MieScat_2d_1 = alloc_vector(Ndim);
847 double* G_MieScat_2d_1;
848 G_MieScat_2d_1 = alloc_vector(Ndim);
849 double* B_MieScat_2d_1;
850 B_MieScat_2d_1 = alloc_vector(Ndim);
851
852 if (0 == MieScat_Sca_Rect_2nd(lz1d, nang, s11_array, refmed, refre, refim,
    radius, R_MieScat_2d_1, G_MieScat_2d_1, B_MieScat_2d_1, param3, Xbar,
    Ybar, Zbar, MaxX, MaxY, Maxk, bw, Absorbance, f_fine, Miec, MieTheoScat))
853 {
854     return 0;
855 }
856 else
857 {
858     FILE* DestMatrixFile;
859     int ret;
860     errno_t err = fopen_s(&DestMatrixFile, "MieScat_destmatrix.txt", "w");
861
862     if (err != 0)
863     {
864         return 0;
865     }
866     else
867     {
868         ret = fprintf_s(DestMatrixFile, "%d\n", Ndim);
869         for (i = 0; i < Ndim; i++)
870         {
871             ret = fprintf_s(DestMatrixFile, "%8.10le\n", R_MieScat_2d_1[i]);
872         }
873
874         for (i = 0; i < Ndim; i++)
875         {
876             ret = fprintf_s(DestMatrixFile, "%8.10le\n", G_MieScat_2d_1[i]);
877         }
878
879         for (i = 0; i < Ndim; i++)
880         {
881             ret = fprintf_s(DestMatrixFile, "%8.10le\n", B_MieScat_2d_1[i]);
882         }
883         fclose(DestMatrixFile);
884
885         return 1;
886     }
887 }
888 }
889
890
891
892 if (TypeCalc[0] == 'N')
893 {
894     int NwINBot;
895
896     FILE* InputFileName;
897     if ((fopen_s(&InputFileName, "Mie_para.txt", "r")) != 0) return 0;
898     for (i = 0; i <= 29; i++)
899     {
900         fscanf_s(InputFileName, "%lf", &dummy);
901         param3[i] = dummy;
902     }
903     fscanf_s(InputFileName, "%d", &FlagCliff);
904     for (i = 1; i <= 70; i++)
905     {
906         fscanf_s(InputFileName, "%lf", &dummy);
907         Xbar[i] = dummy;

```

```
908     }
909     for (i = 1; i <= 70; i++)
910     {
911         fscanf_s(InputFileName, "%lf", &dummy);
912         Ybar[i] = dummy;
913     }
914     for (i = 1; i <= 70; i++)
915     {
916         fscanf_s(InputFileName, "%lf", &dummy);
917         Zbar[i] = dummy;
918     }
919     fscanf_s(InputFileName, "%d", &MaxX);
920     fscanf_s(InputFileName, "%d", &MaxY);
921     fscanf_s(InputFileName, "%d", &Maxk);
922     for (i = 1; i <= 70; i++)
923     {
924         fscanf_s(InputFileName, "%lf", &dummy);
925         bw[i] = dummy;
926     }
927     for (i = 1; i <= 70; i++)
928     {
929         fscanf_s(InputFileName, "%lf", &dummy);
930         Absorbance[i] = dummy;
931     }
932     for (i = 1; i <= 70; i++)
933     {
934         fscanf_s(InputFileName, "%lf", &dummy);
935         f_fine[i] = dummy;
936     }
937     fscanf_s(InputFileName, "%lf", &Miec);
938     for (i = 1; i <= 70; i++)
939     {
940         fscanf_s(InputFileName, "%lf", &dummy);
941         MieTheoScat[i] = dummy;
942     }
943     fscanf_s(InputFileName, "%d", &NwINBot);
944
945     double* lz1d;
946     lz1d = alloc_vector(NwINBot);
947
948     for (i = 0; i < NwINBot; i++)
949     {
950         fscanf_s(InputFileName, "%lf", &dummy);
951         lz1d[i] = dummy;
952     }
953
954     fclose(InputFileName);
955
956     FactorDC = param3[0];
957     gzai = param3[1];
958     Sigma = param3[2];
959     delta = param3[3];
960     deltaLamda = param3[4];
961     r1 = param3[5];
962     d = param3[6];
963     Xgrid = param3[7];
964     Ygrid = param3[8];
965     FacDetailed = param3[9];
966     LakeShoreEPS = param3[10];
967     fai77 = param3[11];
968     theta77 = param3[12];
969     n12 = param3[13];
970     xobs = param3[14];
971     yobs = param3[15];
```



```

972     zobs = param3[16];
973     k1 = param3[17];
974     Km = param3[18];
975     SettleShodo = param3[19];
976     TheoShodo = param3[20];
977     Rittaikaku = param3[21];
978     refl = param3[22];
979     delta0 = param3[23];
980     FactorZ = param3[24];
981     FlagStepAgainstSlope = param3[25];
982     InitZ = param3[26];
983     theta1 = param3[27];
984     iPetzoldPF = param3[28];
985     y0 = param3[29];
986
987     Ndim = (MaxX * static_cast<int>(FacDetailed) + 1) * (MaxY * static_cast<int>(FacDetailed) + 1);
988
989     double* R_Mie_2d_1;
990     R_Mie_2d_1 = alloc_vector(Ndim);
991     double* G_Mie_2d_1;
992     G_Mie_2d_1 = alloc_vector(Ndim);
993     double* B_Mie_2d_1;
994     B_Mie_2d_1 = alloc_vector(Ndim);
995
996     if (0 == MieSca_Rect_2nd(lz1d, R_Mie_2d_1, G_Mie_2d_1, B_Mie_2d_1, param3,
997         Xbar, Ybar, Zbar, MaxX, MaxY, Maxk, bw, Absorbance, f_fine, Miec,
998         MieTheoScat))
999     {
1000         return 0;
1001     }
1002     else
1003     {
1004         FILE* DestMatrixFile;
1005         int ret;
1006         errno_t err = fopen_s(&DestMatrixFile, "Mie_destmatrix.txt", "w");
1007
1008         if (err != 0)
1009         {
1010             return 0;
1011         }
1012         else
1013         {
1014             ret = fprintf_s(DestMatrixFile, "%d¥n", Ndim);
1015             for (i = 0; i < Ndim; i++)
1016             {
1017                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", R_Mie_2d_1[i]);
1018             }
1019             for (i = 0; i < Ndim; i++)
1020             {
1021                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", G_Mie_2d_1[i]);
1022             }
1023             for (i = 0; i < Ndim; i++)
1024             {
1025                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", B_Mie_2d_1[i]);
1026             }
1027             fclose(DestMatrixFile);
1028             return 1;
1029         }
1030     }
1031 }
1032

```

```
1033
1034     if (TypeCalc[0] == 'F')
1035     {
1036         int NwINBot;
1037
1038         FILE* InputFileName;
1039         if ((fopen_s(&InputFileName, "ES_para.txt", "r")) != 0) return 0;
1040         for (i = 0; i <= 28; i++)
1041         {
1042             fscanf_s(InputFileName, "%lf", &dummy);
1043             param3[i] = dummy;
1044         }
1045         fscanf_s(InputFileName, "%d", &FlagCliff);
1046         for (i = 1; i <= 70; i++)
1047         {
1048             fscanf_s(InputFileName, "%lf", &dummy);
1049             Xbar[i] = dummy;
1050         }
1051         for (i = 1; i <= 70; i++)
1052         {
1053             fscanf_s(InputFileName, "%lf", &dummy);
1054             Ybar[i] = dummy;
1055         }
1056         for (i = 1; i <= 70; i++)
1057         {
1058             fscanf_s(InputFileName, "%lf", &dummy);
1059             Zbar[i] = dummy;
1060         }
1061         fscanf_s(InputFileName, "%d", &MaxX);
1062         fscanf_s(InputFileName, "%d", &MaxY);
1063         fscanf_s(InputFileName, "%d", &Maxk);
1064         for (i = 1; i <= 70; i++)
1065         {
1066             fscanf_s(InputFileName, "%lf", &dummy);
1067             bw[i] = dummy;
1068         }
1069         for (i = 1; i <= 70; i++)
1070         {
1071             fscanf_s(InputFileName, "%lf", &dummy);
1072             Absorbance[i] = dummy;
1073         }
1074         for (i = 1; i <= 70; i++)
1075         {
1076             fscanf_s(InputFileName, "%lf", &dummy);
1077             f_fine[i] = dummy;
1078         }
1079         fscanf_s(InputFileName, "%lf", &Miec);
1080         for (i = 1; i <= 70; i++)
1081         {
1082             fscanf_s(InputFileName, "%lf", &dummy);
1083             MieTheoScat[i] = dummy;
1084         }
1085         fscanf_s(InputFileName, "%d", &NwINBot);
1086
1087         double* lz1d;
1088         lz1d = alloc_vector(NwINBot);
1089
1090         for (i = 0; i < NwINBot; i++)
1091         {
1092             fscanf_s(InputFileName, "%lf", &dummy);
1093             lz1d[i] = dummy;
1094         }
1095
1096         fclose(InputFileName);
```

```

1097
1098     FactorDC = param3[0];
1099     gzai = param3[1];
1100     Sigma = param3[2];
1101     delta = param3[3];
1102     deltaLamda = param3[4];
1103     r1 = param3[5];
1104     d = param3[6];
1105     Xgrid = param3[7];
1106     Ygrid = param3[8];
1107     FacDetailed = param3[9];
1108     LakeShoreEPS = param3[10];
1109     fai77 = param3[11];
1110     theta77 = param3[12];
1111     n12 = param3[13];
1112     xobs = param3[14];
1113     yobs = param3[15];
1114     zobs = param3[16];
1115     k1 = param3[17];
1116     Km = param3[18];
1117     SettleShodo = param3[19];
1118     TheoShodo = param3[20];
1119     Rittaiaku = param3[21];
1120     refl = param3[22];
1121     delta0 = param3[23];
1122     FactorZ = param3[24];
1123     FlagStepAgainstSlope = param3[25];
1124     InitZ = param3[26];
1125     theta1 = param3[27];
1126     y0 = param3[28];
1127
1128     Ndim = (MaxX * static_cast<int>(FacDetailed) + 1) * (MaxY * static_cast<int>(
1129         (FacDetailed) + 1);
1130
1131     double* R_ES_2d_1;
1132     R_ES_2d_1 = alloc_vector(Ndim);
1133     double* G_ES_2d_1;
1134     G_ES_2d_1 = alloc_vector(Ndim);
1135     double* B_ES_2d_1;
1136     B_ES_2d_1 = alloc_vector(Ndim);
1137
1138     if (0 == DenstFlucSca_Rect_2nd(lz1d, R_ES_2d_1, G_ES_2d_1, B_ES_2d_1,
1139         param3, Xbar, Ybar, Zbar, MaxX, MaxY, Maxk, bw, Absorbance, f_fine, Miec,
1140         MieTheoScat))
1141     {
1142         return 0;
1143     }
1144     else
1145     {
1146         FILE* DestMatrixFile;
1147         int ret;
1148         errno_t err = fopen_s(&DestMatrixFile, "ES_destmatrix.txt", "w");
1149
1150         if (err != 0)
1151         {
1152             return 0;
1153         }
1154         else
1155         {
1156             ret = fprintf_s(DestMatrixFile, "%d¥n", Ndim);
1157             for (i = 0; i < Ndim; i++)
1158             {
1159                 ret = fprintf_s(DestMatrixFile, "%8.10le¥n", R_ES_2d_1[i]);
1160             }
1161         }
1162     }

```

```

1158
1159         for (i = 0; i < Ndim; i++)
1160         {
1161             ret = fprintf_s(DestMatrixFile, "%8.10le\n", G_ES_2d_1[i]);
1162         }
1163
1164         for (i = 0; i < Ndim; i++)
1165         {
1166             ret = fprintf_s(DestMatrixFile, "%8.10le\n", B_ES_2d_1[i]);
1167         }
1168         fclose(DestMatrixFile);
1169         return 1;
1170     }
1171 }
1172 }
1173
1174
1175 if (TypeCalc[0] == 'D')
1176 {
1177     int NwINBot;
1178
1179     FILE* InputFileName;
1180     if ((fopen_s(&InputFileName, "Bot_para.txt", "r")) != 0) return 0;
1181     for (i = 0; i <= 28; i++)
1182     {
1183         fscanf_s(InputFileName, "%lf", &dummy);
1184         param3[i] = dummy;
1185     }
1186     fscanf_s(InputFileName, "%d", &FlagCliff);
1187     for (i = 1; i <= 70; i++)
1188     {
1189         fscanf_s(InputFileName, "%lf", &dummy);
1190         Xbar[i] = dummy;
1191     }
1192     for (i = 1; i <= 70; i++)
1193     {
1194         fscanf_s(InputFileName, "%lf", &dummy);
1195         Ybar[i] = dummy;
1196     }
1197     for (i = 1; i <= 70; i++)
1198     {
1199         fscanf_s(InputFileName, "%lf", &dummy);
1200         Zbar[i] = dummy;
1201     }
1202     fscanf_s(InputFileName, "%d", &MaxX);
1203     fscanf_s(InputFileName, "%d", &MaxY);
1204     for (i = 1; i <= 70; i++)
1205     {
1206         fscanf_s(InputFileName, "%lf", &dummy);
1207         bw[i] = dummy;
1208     }
1209     for (i = 1; i <= 70; i++)
1210     {
1211         fscanf_s(InputFileName, "%lf", &dummy);
1212         Absorbance[i] = dummy;
1213     }
1214     for (i = 1; i <= 70; i++)
1215     {
1216         fscanf_s(InputFileName, "%lf", &dummy);
1217         f_fine[i] = dummy;
1218     }
1219     fscanf_s(InputFileName, "%lf", &Miec);
1220     for (i = 1; i <= 70; i++)
1221     {

```

```

1222         fscanf_s(InputFileName, "%lf", &dummy);
1223         MieTheoScat[i] = dummy;
1224     }
1225     fscanf_s(InputFileName, "%d", &NwINBot);
1226
1227     double* lz1d;
1228     lz1d = alloc_vector(NwINBot);
1229
1230     for (i = 0; i < NwINBot; i++)
1231     {
1232         fscanf_s(InputFileName, "%lf", &dummy);
1233         lz1d[i] = dummy;
1234     }
1235
1236     fclose(InputFileName);
1237
1238     FactorDC = param3[0];
1239     gzai = param3[1];
1240     Sigma = param3[2];
1241     delta = param3[3];
1242     deltaLamda = param3[4];
1243     r1 = param3[5];
1244     d = param3[6];
1245     Xgrid = param3[7];
1246     Ygrid = param3[8];
1247     FacDetailed = param3[9];
1248     LakeShoreEPS = param3[10];
1249     fai77 = param3[11];
1250     theta77 = param3[12];
1251     n12 = param3[13];
1252     xobs = param3[14];
1253     yobs = param3[15];
1254     zobs = param3[16];
1255     k1 = param3[17];
1256     Km = param3[18];
1257     SettleShodo = param3[19];
1258     TheoShodo = param3[20];
1259     Rittaikaku = param3[21];
1260     refl = param3[22];
1261     FlagStepAgainstSlope = param3[23];
1262     InitZ = param3[24];
1263     theta1 = param3[25];
1264     y0 = param3[26];
1265     iLambertian = param3[27];
1266     FactorZ = param3[28];
1267
1268     Ndim = (MaxX * static_cast<int>(FacDetailed) + 1) * (MaxY * static_cast<int>(
        (FacDetailed) + 1);
1269
1270     double* R_Bot_2d_1;
1271     R_Bot_2d_1 = alloc_vector(Ndim);
1272     double* G_Bot_2d_1;
1273     G_Bot_2d_1 = alloc_vector(Ndim);
1274     double* B_Bot_2d_1;
1275     B_Bot_2d_1 = alloc_vector(Ndim);
1276
1277     if (0 == IrrRefBottom_Rect_2nd(lz1d, R_Bot_2d_1, G_Bot_2d_1, B_Bot_2d_1,
        param3, Xbar, Ybar, Zbar, MaxX, MaxY, bw, Absorbance, f_fine, Miec,
        MieTheoScat))
1278     {
1279         return 0;
1280     }
1281     else
1282     {

```

```

1283     FILE* DestMatrixFile;
1284     int ret;
1285     errno_t err = fopen_s(&DestMatrixFile, "Bot_destmatrix.txt", "w");
1286
1287     if (err != 0)
1288     {
1289         return 0;
1290     }
1291     else
1292     {
1293         ret = fprintf_s(DestMatrixFile, "%d\n", Ndim);
1294         for (i = 0; i < Ndim; i++)
1295         {
1296             ret = fprintf_s(DestMatrixFile, "%8.10le\n", R_Bot_2d_1[i]);
1297         }
1298
1299         for (i = 0; i < Ndim; i++)
1300         {
1301             ret = fprintf_s(DestMatrixFile, "%8.10le\n", G_Bot_2d_1[i]);
1302         }
1303
1304         for (i = 0; i < Ndim; i++)
1305         {
1306             ret = fprintf_s(DestMatrixFile, "%8.10le\n", B_Bot_2d_1[i]);
1307         }
1308         fclose(DestMatrixFile);
1309         return 1;
1310     }
1311 }
1312 }
1313
1314
1315 if (TypeCalc[0] == 'B')
1316 {
1317     FILE* InputFileName;
1318     if ((fopen_s(&InputFileName, "Bot_para.txt", "r")) != 0) return 0;
1319     for (i = 0; i <= 24; i++)
1320     {
1321         fscanf_s(InputFileName, "%lf", &dummy);
1322         param3[i] = dummy;
1323     }
1324     fscanf_s(InputFileName, "%d", &FlagCliff);
1325     for (i = 1; i <= 70; i++)
1326     {
1327         fscanf_s(InputFileName, "%lf", &dummy);
1328         Xbar[i] = dummy;
1329     }
1330     for (i = 1; i <= 70; i++)
1331     {
1332         fscanf_s(InputFileName, "%lf", &dummy);
1333         Ybar[i] = dummy;
1334     }
1335     for (i = 1; i <= 70; i++)
1336     {
1337         fscanf_s(InputFileName, "%lf", &dummy);
1338         Zbar[i] = dummy;
1339     }
1340     fscanf_s(InputFileName, "%d", &MaxX);
1341     fscanf_s(InputFileName, "%d", &MaxY);
1342     for (i = 1; i <= 70; i++)
1343     {
1344         fscanf_s(InputFileName, "%lf", &dummy);
1345         bw[i] = dummy;
1346     }

```

```

1347         for (i = 1; i <= 70; i++)
1348         {
1349             fscanf_s(InputFileName, "%lf", &dummy);
1350             Absorbance[i] = dummy;
1351         }
1352         for (i = 1; i <= 70; i++)
1353         {
1354             fscanf_s(InputFileName, "%lf", &dummy);
1355             f_fine[i] = dummy;
1356         }
1357         fscanf_s(InputFileName, "%lf", &Miec);
1358         fclose(InputFileName);
1359
1360         FactorDC = param3[0];
1361         gzai = param3[1];
1362         Sigma = param3[2];
1363         delta = param3[3];
1364         deltaLamda = param3[4];
1365         r1 = param3[5];
1366         d = param3[6];
1367         Xgrid = param3[7];
1368         Ygrid = param3[8];
1369         FacDetailed = param3[9];
1370         LakeShoreEPS = param3[10];
1371         fai77 = param3[11];
1372         theta77 = param3[12];
1373         n12 = param3[13];
1374         xobs = param3[14];
1375         yobs = param3[15];
1376         zobs = param3[16];
1377         k1 = param3[17];
1378         Km = param3[18];
1379         SettleShodo = param3[19];
1380         TheoShodo = param3[20];
1381         Rittaiaku = param3[21];
1382         refl = param3[22];
1383         iLambertian = param3[24];
1384
1385         Ndim = (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1386             1) * (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1387             1);
1388
1389         double* R_Bot_2d_1;
1390         R_Bot_2d_1 = alloc_vector(Ndim);
1391         double* G_Bot_2d_1;
1392         G_Bot_2d_1 = alloc_vector(Ndim);
1393         double* B_Bot_2d_1;
1394         B_Bot_2d_1 = alloc_vector(Ndim);
1395
1396         if (0 == IrrRefBottom(R_Bot_2d_1, G_Bot_2d_1, B_Bot_2d_1, param3, FlagCliff,
1397             Xbar, Ybar, Zbar, MaxX, MaxY, bw, Absorbance, f_fine, Miec))
1398         {
1399             return 0;
1400         }
1401         else
1402         {
1403             FILE* DestMatrixFile;
1404             int ret;
1405             errno_t err = fopen_s(&DestMatrixFile, "Bot_destmatrix.txt", "w");
1406
1407             if (err != 0)
1408             {
1409                 return 0;
1410             }
1411         }

```

```

1408         else
1409         {
1410             ret=fprintf_s(DestMatrixFile, "%d\n", Ndim);
1411             for (i = 0; i < Ndim; i++)
1412             {
1413                 ret = fprintf_s(DestMatrixFile, "%8.10le\n", R_Bot_2d_1[i]);
1414             }
1415             for (i = 0; i < Ndim; i++)
1416             {
1417                 ret = fprintf_s(DestMatrixFile, "%8.10le\n", G_Bot_2d_1[i]);
1418             }
1419             for (i = 0; i < Ndim; i++)
1420             {
1421                 ret = fprintf_s(DestMatrixFile, "%8.10le\n", B_Bot_2d_1[i]);
1422             }
1423             fclose(DestMatrixFile);
1424             return 1;
1425         }
1426     }
1427 }
1428
1429 if (TypeCalc[0] == 'E')
1430 {
1431     FILE* InputFileName;
1432     if ((fopen_s(&InputFileName, "ES_para.txt", "r")) != 0) return 0;
1433     for (i = 0; i <= 25; i++)
1434     {
1435         fscanf_s(InputFileName, "%lf", &dummy);
1436         param3[i] = dummy;
1437     }
1438     fscanf_s(InputFileName, "%d", &FlagCliff);
1439     for (i = 1; i <= 70; i++)
1440     {
1441         fscanf_s(InputFileName, "%lf", &dummy);
1442         Xbar[i] = dummy;
1443     }
1444     for (i = 1; i <= 70; i++)
1445     {
1446         fscanf_s(InputFileName, "%lf", &dummy);
1447         Ybar[i] = dummy;
1448     }
1449     for (i = 1; i <= 70; i++)
1450     {
1451         fscanf_s(InputFileName, "%lf", &dummy);
1452         Zbar[i] = dummy;
1453     }
1454     fscanf_s(InputFileName, "%d", &MaxX);
1455     fscanf_s(InputFileName, "%d", &MaxY);
1456     fscanf_s(InputFileName, "%d", &Maxk);
1457     for (i = 1; i <= 70; i++)
1458     {
1459         fscanf_s(InputFileName, "%lf", &dummy);
1460         bw[i] = dummy;
1461     }
1462     for (i = 1; i <= 70; i++)
1463     {
1464         fscanf_s(InputFileName, "%lf", &dummy);
1465         Absorbance[i] = dummy;
1466     }
1467     for (i = 1; i <= 70; i++)
1468     {
1469         fscanf_s(InputFileName, "%lf", &dummy);
1470     }
1471

```



```

1472         f_fine[i] = dummy;
1473     }
1474     fscanf_s(InputFileName, "%lf", &Miec);
1475     fclose(InputFileName);
1476
1477     FactorDC = param3[0];
1478     gzai = param3[1];
1479     Sigma = param3[2];
1480     delta = param3[3];
1481     deltaLamda = param3[4];
1482     r1 = param3[5];
1483     d = param3[6];
1484     Xgrid = param3[7];
1485     Ygrid = param3[8];
1486     FacDetailed = param3[9];
1487     LakeShoreEPS = param3[10];
1488     fai77 = param3[11];
1489     theta77 = param3[12];
1490     n12 = param3[13];
1491     xobs = param3[14];
1492     yobs = param3[15];
1493     zobs = param3[16];
1494     k1 = param3[17];
1495     Km = param3[18];
1496     SettleShodo = param3[19];
1497     TheoShodo = param3[20];
1498     Rittaikaku = param3[21];
1499     refl = param3[22];
1500     delta0 = param3[23];
1501     FactorZ = param3[24];
1502
1503     Ndim = (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1504             1) * (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1505             1);
1506
1507     double* R_ES_2d_1;
1508     R_ES_2d_1 = alloc_vector(Ndim);
1509     double* G_ES_2d_1;
1510     G_ES_2d_1 = alloc_vector(Ndim);
1511     double* B_ES_2d_1;
1512     B_ES_2d_1 = alloc_vector(Ndim);
1513
1514     if (0 == DenstFlucSca(R_ES_2d_1, G_ES_2d_1, B_ES_2d_1, param3, FlagCliff,
1515                         Xbar, Ybar, Zbar, MaxX, MaxY, Maxk, bw, Absorbance, f_fine, Miec))
1516     {
1517         return 0;
1518     }
1519     else
1520     {
1521         FILE* DestMatrixFile;
1522         errno_t err = fopen_s(&DestMatrixFile, "ES_destmatrix.txt", "w");
1523         if (err != 0)
1524         {
1525             return 0;
1526         }
1527         else
1528         {
1529             fprintf(DestMatrixFile, "%d\n", Ndim);
1530             for (i = 0; i < Ndim; i++)
1531                 fprintf(DestMatrixFile, "%8.10le\n", R_ES_2d_1[i]);
1532             for (i = 0; i < Ndim; i++)
1533                 fprintf(DestMatrixFile, "%8.10le\n", G_ES_2d_1[i]);
1534             for (i = 0; i < Ndim; i++)
1535                 fprintf(DestMatrixFile, "%8.10le\n", B_ES_2d_1[i]);
1536         }
1537     }

```

```
1533         fprintf(DestMatrixFile, "%8.10le\n", B_ES_2d_1[i]);
1534         fclose(DestMatrixFile);
1535         return 1;
1536     }
1537 }
1538 }
1539
1540 if (TypeCalc[0] == 'M')
1541 {
1542     FILE* InputFileName;
1543     if ((fopen_s(&InputFileName, "Mie_para.txt", "r")) != 0) return 0;
1544     for (i = 0; i <= 31; i++)
1545     {
1546         fscanf_s(InputFileName, "%lf", &dummy);
1547         param3[i] = dummy;
1548     }
1549     fscanf_s(InputFileName, "%d", &FlagCliff);
1550     for (i = 1; i <= 70; i++)
1551     {
1552         fscanf_s(InputFileName, "%lf", &dummy);
1553         Xbar[i] = dummy;
1554     }
1555     for (i = 1; i <= 70; i++)
1556     {
1557         fscanf_s(InputFileName, "%lf", &dummy);
1558         Ybar[i] = dummy;
1559     }
1560     for (i = 1; i <= 70; i++)
1561     {
1562         fscanf_s(InputFileName, "%lf", &dummy);
1563         Zbar[i] = dummy;
1564     }
1565     fscanf_s(InputFileName, "%d", &MaxX);
1566     fscanf_s(InputFileName, "%d", &MaxY);
1567     fscanf_s(InputFileName, "%d", &Maxk);
1568     for (i = 1; i <= 70; i++)
1569     {
1570         fscanf_s(InputFileName, "%lf", &dummy);
1571         bw[i] = dummy;
1572     }
1573     for (i = 1; i <= 70; i++)
1574     {
1575         fscanf_s(InputFileName, "%lf", &dummy);
1576         Absorbance[i] = dummy;
1577     }
1578     for (i = 1; i <= 70; i++)
1579     {
1580         fscanf_s(InputFileName, "%lf", &dummy);
1581         f_fine[i] = dummy;
1582     }
1583     fscanf_s(InputFileName, "%lf", &Miec);
1584     fclose(InputFileName);
1585
1586     FactorDC = param3[0];
1587     gzai = param3[1];
1588     Sigma = param3[2];
1589     delta = param3[3];
1590     deltaLamda = param3[4];
1591     r1 = param3[5];
1592     d = param3[6];
1593     Xgrid = param3[7];
1594     Ygrid = param3[8];
1595     FacDetailed = param3[9];
1596     LakeShoreEPS = param3[10];
```

```

1597     fai77 = param3[11];
1598     theta77 = param3[12];
1599     n12 = param3[13];
1600     xobs = param3[14];
1601     yobs = param3[15];
1602     zobs = param3[16];
1603     k1 = param3[17];
1604     Km = param3[18];
1605     SettleShodo = param3[19];
1606     TheoShodo = param3[20];
1607     Rittaiaku = param3[21];
1608     refl = param3[22];
1609     delta0 = param3[23];
1610     FactorZ = param3[24];
1611     Miea0 = param3[25];
1612     Miea1 = param3[26];
1613     Miea2 = param3[27];
1614     Miea3 = param3[28];
1615     Miea4 = param3[29];
1616     iPetzoldPF = param3[30];
1617
1618     Ndim = (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1           1) * (static_cast<int>(2.0 * r1 + 1.0) * static_cast<int>(FacDetailed) +
1           1);
1619
1620     double* R_Mie_2d_1;
1621     R_Mie_2d_1 = alloc_vector(Ndim);
1622     double* G_Mie_2d_1;
1623     G_Mie_2d_1 = alloc_vector(Ndim);
1624     double* B_Mie_2d_1;
1625     B_Mie_2d_1 = alloc_vector(Ndim);
1626
1627     if (0 == MieSca(R_Mie_2d_1, G_Mie_2d_1, B_Mie_2d_1, param3, FlagCliff, Xbar,
1628     Ybar, Zbar, MaxX, MaxY, Maxk, bw, Absorbance, f_fine, Miec))
1629     {
1630         return 0;
1631     }
1632     else
1633     {
1634         FILE* DestMatrixFile;
1635         errno_t err = fopen_s(&DestMatrixFile, "Mie_destmatrix.txt", "w");
1636         if (err != 0)
1637         {
1638             return 0;
1639         }
1640         else
1641         {
1642             fprintf(DestMatrixFile, "%d\n", Ndim);
1643             for (i = 0; i < Ndim; i++)
1644                 fprintf(DestMatrixFile, "%8.10le\n", R_Mie_2d_1[i]);
1645             for (i = 0; i < Ndim; i++)
1646                 fprintf(DestMatrixFile, "%8.10le\n", G_Mie_2d_1[i]);
1647             for (i = 0; i < Ndim; i++)
1648                 fprintf(DestMatrixFile, "%8.10le\n", B_Mie_2d_1[i]);
1649             fclose(DestMatrixFile);
1650             return 1;
1651         }
1652     }
1653 }
1654
1655 if (TypeCalc[0] == 'Q')
1656 {
1657     double MinOki, MaxOki;
1658     double Xrange, Yrange;

```

```

1658
1659 FILE* InputFileName;
1660 if ((fopen_s(&InputFileName, "SplinePre.txt", "r")) != 0) return 0;
1661 for (i = 1; i <= 3; i++)
1662     for (j = 1; j <= 3; j++)
1663     {
1664         fscanf_s(InputFileName, "%lf", &dummy);
1665         HomG[i][j] = dummy;
1666     }
1667 for (i = 1; i <= 3; i++)
1668     for (j = 1; j <= 3; j++)
1669     {
1670         fscanf_s(InputFileName, "%lf", &dummy);
1671         iHomG[i][j] = dummy;
1672     }
1673 fscanf_s(InputFileName, "%d", &ScreenTrans);
1674 fscanf_s(InputFileName, "%d", &ScreenLong);
1675 fscanf_s(InputFileName, "%d", &ScreenOrigTrans);
1676 fscanf_s(InputFileName, "%d", &ScreenOrigLong);
1677 fscanf_s(InputFileName, "%lf", &epsilon);
1678 fscanf_s(InputFileName, "%d", &MaxX);
1679 fscanf_s(InputFileName, "%d", &MaxY);
1680 fscanf_s(InputFileName, "%lf", &FacDetailed);
1681 fscanf_s(InputFileName, "%lf", &MinOk);
1682 fscanf_s(InputFileName, "%lf", &MaxOk);
1683 fscanf_s(InputFileName, "%lf", &Xrange);
1684 fscanf_s(InputFileName, "%lf", &Yrange);
1685 fscanf_s(InputFileName, "%lf", &FactorDC);
1686
1687 long ndim1;
1688 fscanf_s(InputFileName, "%ld", &ndim1);
1689
1690
1691 double** R_x_2d;
1692 R_x_2d = alloc_matrix(ndim1+1, ndim1 + 1);
1693 double** G_x_2d;
1694 G_x_2d = alloc_matrix(ndim1 + 1, ndim1 + 1);
1695 double** B_x_2d;
1696 B_x_2d = alloc_matrix(ndim1 + 1, ndim1 + 1);
1697
1698 for (i = 0; i <= ndim1; i++)
1699     for (j = 0; j <= ndim1; j++)
1700     {
1701         fscanf_s(InputFileName, "%lf", &dummy);
1702         R_x_2d[i][j] = dummy;
1703     }
1704
1705 for (i = 0; i <= ndim1; i++)
1706     for (j = 0; j <= ndim1; j++)
1707     {
1708         fscanf_s(InputFileName, "%lf", &dummy);
1709         G_x_2d[i][j] = dummy;
1710     }
1711
1712 for (i = 0; i <= ndim1; i++)
1713     for (j = 0; j <= ndim1; j++)
1714     {
1715         fscanf_s(InputFileName, "%lf", &dummy);
1716         B_x_2d[i][j] = dummy;
1717     }
1718 fclose(InputFileName);
1719
1720 long ScreenNtrans1Long;
1721 ScreenNtrans1Long = static_cast<long>(ScreenTrans) * static_cast<long>

```

```

(ScreenLong);
1722
1723     double* R88_1;
1724     R88_1 = alloc_vector(ScreenNtrans1Long + 1);
1725     double* G88_1;
1726     G88_1 = alloc_vector(ScreenNtrans1Long + 1);
1727     double* B88_1;
1728     B88_1 = alloc_vector(ScreenNtrans1Long + 1);
1729     int* ScreenFullIFlag_1;
1730     ScreenFullIFlag_1 = alloc_intvector(ScreenNtrans1Long + 1);
1731     double* MaxYY;
1732     MaxYY = alloc_vector(ScreenTrans);
1733     int* T88_1;
1734     T88_1 = alloc_intvector(ScreenNtrans1Long + 1);
1735
1736     for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1737         T88_1[ilong] = 0;
1738
1739     for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1740         ScreenFullIFlag_1[ilong] = 0;
1741
1742     for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1743         R88_1[ilong] = 0.0;
1744
1745     for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1746         G88_1[ilong] = 0.0;
1747
1748     for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1749         B88_1[ilong] = 0.0;
1750
1751     long ilong;
1752
1753     int iCountLL;
1754     iCountLL = 0;
1755
1756     double xtemp, ytemp, xtemp_1, ytemp_1, xtemp_2, ytemp_2;
1757     int ret;
1758
1759     double IntegralLL;
1760
1761     IntegralLL = 0.0;
1762
1763     double x1, y1, x2, y2;
1764
1765     x1 = 100.0;
1766     y1 = 100.0;
1767     x2 = -100.0;
1768     y2 = -100.0;
1769
1770     double x3, y3;
1771     x3 = 0.0;
1772     y3 = Max0ki+0.5;
1773
1774     ret = ConvertPhoto2Circle2(ScreenTrans, ScreenLong, ScreenOrigTrans,
1775                               ScreenOrigLong, false, x3, y3, HomG, iHomG);
1776
1777
1778     for (i = 0; i < ScreenTrans; i++)
1779         for (j = 0; j < ScreenLong; j++)
1780         {
1781             xtemp = static_cast<double>(i);
1782             ytemp = static_cast<double>(ScreenLong-1-j);
1783             if (y3 < ytemp)

```

```

1784     {
1785         ret = ConvertPhoto2Circle2(ScreenTrans, ScreenLong,
                                     ScreenOrigTrans, ScreenOrigLong, true, xtemp, ytemp, HomG,
                                     iHomG);
1786         xtemp_1 = xtemp * cos(epsiron) - ytemp * sin(epsiron);
1787         ytemp_1 = xtemp * sin(epsiron) + ytemp * cos(epsiron);
1788
1789
1790         if ((ytemp_1 > MinOkki) && (ytemp_1 < MaxOkki))
1791         {
1792             T88_1[i * ScreenLong + j] = 1;
1793         }
1794
1795
1796
1797         if (x1 > xtemp_1) x1 = xtemp_1;
1798         if (y1 > ytemp_1) y1 = ytemp_1;
1799         if (x2 < xtemp_1) x2 = xtemp_1;
1800         if (y2 < ytemp_1) y2 = ytemp_1;
1801
1802
1803         for (int i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1+
+)
1804             for (int j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
1805             {
1806                 xtemp = (static_cast<double>(i1) - 0.5 *
static_cast<double>(MaxX) * FacDetailed) /
static_cast<double>(MaxX) * FacDetailed) * Xrange;
1807                 ytemp = (static_cast<double>(j1) - 0.5 *
static_cast<double>(MaxY) * FacDetailed) /
static_cast<double>(MaxY) * FacDetailed) * Yrange;
1808                 xtemp_2 = (static_cast<double>(i1 + 1) - 0.5 *
static_cast<double>(MaxX) * FacDetailed) /
static_cast<double>(MaxX) * FacDetailed) * Xrange;
1809                 ytemp_2 = (static_cast<double>(j1 + 1) - 0.5 *
static_cast<double>(MaxY) * FacDetailed) /
static_cast<double>(MaxY) * FacDetailed) * Yrange;
1810                 if ((xtemp <= xtemp_1) && (xtemp_1 < xtemp_2) && (ytemp
<= ytemp_1) && (ytemp_1 < ytemp_2) && (ytemp_1 >
MinOkki) && (ytemp_1 < MaxOkki))
1811                 {
1812                     R88_1[i * ScreenLong + j] = R_x_2d[i1][j1];
1813                     G88_1[i * ScreenLong + j] = G_x_2d[i1][j1];
1814                     B88_1[i * ScreenLong + j] = B_x_2d[i1][j1];
1815                     ScreenFullFlag_1[i * ScreenLong + j] = 1;
1816                     IntegralLL = IntegralLL + 0.2126 * R88_1[i *
ScreenLong + j] + 0.7152 * G88_1[i * ScreenLong +
j] + 0.0722 * B88_1[i * ScreenLong + j];
1817                 }
1818             }
1819         }
1820     }
1821 }
1822
1823 IntegralLL = IntegralLL / ((fabs(2.0 * x2) + fabs(2.0 * x1)) * fabs(y2 -
y1) * 0.5)/FactorDC;
1824
1825 FILE* DestMatrixFile;
1826 errno_t err = fopen_s(&DestMatrixFile, "SplinePreDest.txt", "w");
1827 if (err != 0)
1828 {
1829     return 0;
1830 }

```

```

1831         else
1832         {
1833             fprintf(DestMatrixFile, "%ld\n", ScreenNtrans1Long);
1834             for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1835                 fprintf(DestMatrixFile, "%8.10le\n", R88_1[ilong]);
1836             for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1837                 fprintf(DestMatrixFile, "%8.10le\n", G88_1[ilong]);
1838             for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1839                 fprintf(DestMatrixFile, "%8.10le\n", B88_1[ilong]);
1840             for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1841                 fprintf(DestMatrixFile, "%d\n", ScreenFullIFlag_1[ilong]);
1842             for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1843                 fprintf(DestMatrixFile, "%d\n", T88_1[ilong]);
1844
1845             fprintf(DestMatrixFile, "%8.10le\n", IntegralLL);
1846             fclose(DestMatrixFile);
1847
1848             return 1;
1849         }
1850     }
1851 }
1852
1853
1854 if (TypeCalc[0] == 'P')
1855 {
1856     FILE* InputFileName;
1857     if ((fopen_s(&InputFileName, "SplinePre.txt", "r")) != 0) return 0;
1858     for (i = 1; i <= 3; i++)
1859         for (j = 1; j <= 3; j++)
1860         {
1861             fscanf_s(InputFileName, "%lf", &dummy);
1862             HomG[i][j] = dummy;
1863         }
1864     for (i = 1; i <= 3; i++)
1865         for (j = 1; j <= 3; j++)
1866         {
1867             fscanf_s(InputFileName, "%lf", &dummy);
1868             iHomG[i][j] = dummy;
1869         }
1870     fscanf_s(InputFileName, "%d", &ScreenTrans);
1871     fscanf_s(InputFileName, "%d", &ScreenLong);
1872     fscanf_s(InputFileName, "%d", &ScreenOrigTrans);
1873     fscanf_s(InputFileName, "%d", &ScreenOrigLong);
1874     fscanf_s(InputFileName, "%lf", &epsiron);
1875     fscanf_s(InputFileName, "%lf", &r1);
1876     fscanf_s(InputFileName, "%lf", &FacDetailed);
1877
1878     double** R_x_2d;
1879     R_x_2d = alloc_matrix(int((2.0 * r1 + 1.0) * FacDetailed) + 1, int((2.0 *
1880         r1 + 1.0) * FacDetailed) + 1);
1881     double** G_x_2d;
1882     G_x_2d = alloc_matrix(int((2.0 * r1 + 1.0) * FacDetailed) + 1, int((2.
1883         0 * r1 + 1.0) * FacDetailed) + 1);
1884     double** B_x_2d;
1885     B_x_2d = alloc_matrix(int((2.0 * r1 + 1.0) * FacDetailed) + 1, int((2.
1886         0 * r1 + 1.0) * FacDetailed) + 1);
1887
1888     for (i=0; i<=int((2.0*r1+1.0)*FacDetailed); i++)
1889         for (j = 0; j<= int((2.0 * r1 + 1.0) * FacDetailed); j++)
1890         {
1891             fscanf_s(InputFileName, "%lf", &dummy);
1892             R_x_2d[i][j] = dummy;
1893         }
1894     }

```

```

1892         for (i = 0; i <= int((2.0 * r1 + 1.0) * FacDetailed); i++)
1893             for (j = 0; j <= int((2.0 * r1 + 1.0) * FacDetailed); j++)
1894             {
1895                 fscanf_s(InputFileName, "%lf", &dummy);
1896                 G_x_2d[i][j] = dummy;
1897             }
1898
1899         for (i = 0; i <= int((2.0 * r1 + 1.0) * FacDetailed); i++)
1900             for (j = 0; j <= int((2.0 * r1 + 1.0) * FacDetailed); j++)
1901             {
1902                 fscanf_s(InputFileName, "%lf", &dummy);
1903                 B_x_2d[i][j] = dummy;
1904             }
1905         fclose(InputFileName);
1906
1907         long ScreenNtrans1Long;
1908         ScreenNtrans1Long = static_cast<long>(ScreenTrans) * static_cast<long>(ScreenLong);
1909
1910         double* R88_1;
1911         R88_1 = alloc_vector(ScreenNtrans1Long+1);
1912         double* G88_1;
1913         G88_1 = alloc_vector(ScreenNtrans1Long+1);
1914         double* B88_1;
1915         B88_1 = alloc_vector(ScreenNtrans1Long+1);
1916         int* ScreenFullIFlag_1;
1917         ScreenFullIFlag_1 = alloc_intvector(ScreenNtrans1Long+1);
1918         double* MaxYY;
1919         MaxYY = alloc_vector(ScreenTrans);
1920         int* T88_1;
1921         T88_1 = alloc_intvector(ScreenNtrans1Long + 1);
1922
1923         for (long ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1924             T88_1[ilong] = 1;
1925
1926         double MinX1, MaxX1, MinY1, MaxY1;
1927
1928         long ilong;
1929
1930
1931
1932         if (0 == PreSplineLake64(T88_1, HomG, iHomG, ScreenTrans, ScreenLong,
1933             ScreenOrigTrans, ScreenOrigLong, 0, epsilon, r1, FacDetailed, R_x_2d,
1934             G_x_2d, B_x_2d, R88_1, G88_1, B88_1, ScreenFullIFlag_1, MaxYY, MinX1,
1935             MaxX1, MinY1, MaxY1))
1936         {
1937             return 0;
1938         }
1939         else
1940         {
1941             FILE* DestMatrixFile;
1942             errno_t err = fopen_s(&DestMatrixFile, "SplinePreDest.txt", "w");
1943             if (err != 0)
1944             {
1945                 return 0;
1946             }
1947             else
1948             {
1949                 fprintf(DestMatrixFile, "%ld\n", ScreenNtrans1Long);
1950                 for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1951                     fprintf(DestMatrixFile, "%8.10le\n", R88_1[ilong]);
1952                 for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)
1953                     fprintf(DestMatrixFile, "%8.10le\n", G88_1[ilong]);
1954                 for (ilong = 0; ilong <= ScreenNtrans1Long; ilong++)

```



```

1952         fprintf(DestMatrixFile, "%8.10le\n", B88_1[iLong]);
1953         for (iLong = 0; iLong <= ScreenNtrans1Long; iLong++)
1954             fprintf(DestMatrixFile, "%d\n", ScreenFullIFlag_1[iLong]);
1955         for (iLong = 0; iLong <= ScreenNtrans1Long; iLong++)
1956             fprintf(DestMatrixFile, "%d\n", T88_1[iLong]);
1957         for (i = 0; i < ScreenTrans; i++)
1958             fprintf(DestMatrixFile, "%8.10le\n", MaxYY[i]);
1959         fprintf(DestMatrixFile, "%8.10le\n", MinX1);
1960         fprintf(DestMatrixFile, "%8.10le\n", MaxX1);
1961         fprintf(DestMatrixFile, "%8.10le\n", MinY1);
1962         fprintf(DestMatrixFile, "%8.10le\n", MaxY1);
1963         fclose(DestMatrixFile);
1964
1965         return 1;
1966     }
1967 }
1968 }
1969
1970 if (TypeCalc[0] == 'S')
1971 {
1972     long ScreenNtrans1Long;
1973     int ScreenTrans;
1974     int ScreenLong;
1975     long iLong;
1976
1977     double* R88_1;
1978
1979     int* ScreenFullIFlag_1;
1980
1981     double dummy;
1982     int dummy1;
1983
1984     double smax_thre;
1985     long MaxIter;
1986
1987     FILE* InputFileName;
1988     if ((fopen_s(&InputFileName, "SplineSource.txt", "r")) != 0) return 0;
1989     fscanf_s(InputFileName, "%d", &ScreenTrans);
1990     fscanf_s(InputFileName, "%d", &ScreenLong);
1991
1992     ScreenNtrans1Long = static_cast<long>(ScreenTrans) * static_cast<long>(ScreenLong);
1993     ScreenFullIFlag_1 = alloc_intvector(ScreenNtrans1Long + 1);
1994
1995
1996
1997     R88_1 = alloc_vector(ScreenNtrans1Long + 1);
1998
1999     for (iLong = 0; iLong <= ScreenNtrans1Long; iLong++)
2000     {
2001         fscanf_s(InputFileName, "%lf", &dummy);
2002         R88_1[iLong] = dummy;
2003     }
2004     for (iLong = 0; iLong <= ScreenNtrans1Long; iLong++)
2005     {
2006         fscanf_s(InputFileName, "%d", &dummy1);
2007         ScreenFullIFlag_1[iLong] = dummy1;
2008     }
2009     fscanf_s(InputFileName, "%lf", &smax_thre);
2010     fscanf_s(InputFileName, "%ld", &MaxIter);
2011     fclose(InputFileName);
2012
2013     int nFlag;
2014

```

```

2015
2016     if (0 == splineDLL64(ScreenTrans, ScreenLong, R88_1, ScreenFullFlag_1,
2017         {
2018             nFlag = 0;
2019             FILE* DestMatrixFile;
2020             errno_t err = fopen_s(&DestMatrixFile, "SplineResult.txt", "w");
2021             if (err != 0)
2022             {
2023                 std::cout << "Press any key to continue...";
2024                 std::cin.get();
2025                 return 0;
2026             }
2027             else
2028             {
2029                 fprintf(DestMatrixFile, "%ld\n", nFlag);
2030                 fclose(DestMatrixFile);
2031             }
2032             return 0;
2033         }
2034     else
2035     {
2036         nFlag = 1;
2037         FILE* DestMatrixFile;
2038         errno_t err = fopen_s(&DestMatrixFile, "SplineDest.txt", "w");
2039         if (err != 0)
2040         {
2041             std::cout << "Press any key to continue...";
2042             std::cin.get();
2043             return 0;
2044         }
2045         else
2046         {
2047             fprintf(DestMatrixFile, "%ld\n", ScreenTrans1Long);
2048             for (iLong = 0; iLong <= ScreenTrans1Long; iLong++)
2049                 fprintf(DestMatrixFile, "%8.10le\n", R88_1[iLong]);
2050             fclose(DestMatrixFile);
2051
2052             errno_t err = fopen_s(&DestMatrixFile, "SplineResult.txt", "w");
2053             if (err != 0)
2054             {
2055                 std::cout << "Press any key to continue...";
2056                 std::cin.get();
2057                 return 0;
2058             }
2059             else
2060             {
2061                 fprintf(DestMatrixFile, "%ld\n", nFlag);
2062                 fclose(DestMatrixFile);
2063             }
2064             return 1;
2065         }
2066     }
2067 }
2068 }
2069 }
2070
2071 int MieScat_Sca_Rect_2nd(double* IZ1d, int nang, double** s11_table, double refmed,
2072     double refre, double refim, double radius, double* R_Mie_2d_1, double* G_Mie_2d_1,
2073     double* B_Mie_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int
2074     MaxX, int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double
2075     MieC, double* MieScatC_array)
2076 {
2077     int i, j;

```

```
2074     double x11, y11, z11;
2075     double Rtemp, Gtemp, Btemp;
2076     bool Hitpath0;
2077     double scale1;
2078     double Resultvector1[5];
2079     double b11x, b11y, b11z;
2080     double x33, y33;
2081     double s2;
2082     double x4888, y4888;
2083     double thetadash7;
2084     double reffactor22;
2085     double Pi;
2086     Pi = 3.14159265358979;
2087
2088     double alfa, beta;
2089     double Xsum, Ysum, Zsum;
2090     int n;
2091     double Exp1;
2092     double R_val, G_val, B_val;
2093     long i1, j1;
2094     double xtemp, ytemp;
2095     double Gauss;
2096     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,
2097         FacDetailed, LakeShoreEPS;
2098     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,
2099         Rittaikaku, refl;
2100     int k;
2101     double zz;
2102     double FactorZ;
2103     double CosScatAng, delta0, PhaseFunc, L3, L4;
2104     double ScatAng;
2105     double FlagStepAgainstSlope, InitZ, theta1;
2106     double Xrange, Yrange;
2107
2108     double Lrad, Eobs;
2109
2110     double iPetzoldPF;
2111
2112     FactorDC = Params1[0];
2113     gzai = Params1[1];
2114     sigma = Params1[2];
2115     delta = Params1[3];
2116     deltaLamda = Params1[4];
2117     r1 = Params1[5];
2118     d = Params1[6];
2119     Xgrid = Params1[7];
2120     Ygrid = Params1[8];
2121     FacDetailed = Params1[9];
2122     LakeShoreEPS = Params1[10];
2123     fai77 = Params1[11];
2124     theta77 = Params1[12];
2125     n12 = Params1[13];
2126     xobs = Params1[14];
2127     yobs = Params1[15];
2128     zobs = Params1[16];
2129     k1 = Params1[17];
2130     Km = Params1[18];
2131     SettleShodo = Params1[19];
2132     TheoShodo = Params1[20];
2133     Rittaikaku = Params1[21];
2134     refl = Params1[22];
2135     delta0 = Params1[23];
2136     FactorZ = Params1[24];
2137     FlagStepAgainstSlope = Params1[25];
```

```

2136 InitZ = Params1[26];
2137 theta1 = Params1[27];
2138 iPetzoldPF = Params1[28];
2139
2140 Xrange = MaxX * Xgrid;
2141 Yrange = MaxY * Ygrid;
2142
2143 double** R_Mie_2d;
2144 double** G_Mie_2d;
2145 double** B_Mie_2d;
2146
2147 double dS;
2148
2149 double epsilon;
2150 epsilon = atan2(xobs, yobs);
2151 double x111, y111;
2152
2153 double bS;
2154
2155
2156 double dang;
2157 short i_ang;
2158 double TempAng;
2159
2160 dang = Pi / 2.0 / (static_cast<double>(nang - 1));
2161
2162 double* SumPhaseFunc_array;
2163 SumPhaseFunc_array = alloc_vector(71);
2164
2165 double wavel;
2166
2167 double* s11;
2168 s11 = alloc_vector(2 * nang + 10);
2169
2170 int i2;
2171
2172 double s11_val;
2173
2174 for (n = 1; n <= 70; n++)
2175 {
2176     wavel = 380.0 + (static_cast<double>(n - 1)) * deltaLamda;
2177
2178     SumPhaseFunc_array[n] = 0.0;
2179
2180     for (i2 = 1; i2 <= 2 * nang - 1; i2++)
2181         s11[i2] = s11_table[n][i2];
2182
2183
2184     for (i_ang = 0; i_ang <= 2 * nang - 1; i_ang++)
2185     {
2186         if (i_ang == 0) TempAng = 0.0 + 0.1 / 180.0 * Pi;
2187         else TempAng = 0.0 + dang * (static_cast<double>(i_ang));
2188
2189         s11_val = BHMIE1_fromTable(TempAng, s11, nang);
2190
2191         SumPhaseFunc_array[n] = SumPhaseFunc_array[n] + 2.0 * Pi * sin(
2192             TempAng) * dang * s11_val;
2193     }
2194 }
2195
2196 R_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
2197     static_cast<int>(FacDetailed) + 1);
2198 G_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *

```

```

static_cast<int>(FacDetailed) + 1);
2198 B_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2199
2200 for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2201     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2202     {
2203         R_Mie_2d[i1][j1] = 0.0;
2204         G_Mie_2d[i1][j1] = 0.0;
2205         B_Mie_2d[i1][j1] = 0.0;
2206     }
2207
2208 for (i1 = 0; i1 <= (MaxX * static_cast<int>(FacDetailed) + 0) * (MaxY *
static_cast<int>(FacDetailed) + 0); i1++)
2209 {
2210     R_Mie_2d_1[i1] = 0.0;
2211     G_Mie_2d_1[i1] = 0.0;
2212     B_Mie_2d_1[i1] = 0.0;
2213 }
2214
2215 for (i = 0; i <= MaxX; i++)
2216     for (j = 0; j <= MaxY; j++)
2217         for (k = 0; k <= Maxk; k++)
2218         {
2219             x11 = (static_cast<double>(i) - 0.5 * static_cast<double>(MaxX)) /
static_cast<double>(MaxX) * Xrange;
2220             y11 = (static_cast<double>(j) - 0.5 * static_cast<double>(MaxY)) /
static_cast<double>(MaxY) * Yrange;
2221
2222             x111 = x11 * cos(epsiron) + y11 * sin(epsiron);
2223             y111 = -x11 * sin(epsiron) + y11 * cos(epsiron);
2224
2225             x11 = x111;
2226             y11 = y111;
2227
2228             if (FlagStepAgainstSlope == 1.0)
2229             {
2230                 z11 = -d;
2231                 theta1 = 0.0;
2232             }
2233             else
2234             {
2235                 z11 = -(static_cast<double>(j) / static_cast<double>(MaxY) *
Yrange) * tan(theta1) - InitZ;
2236                 d = InitZ + Yrange * tan(theta1);
2237             }
2238
2239             zz = -double(k) * FactorZ;
2240
2241             if ((z11 - zz > -LakeShoreEPS) || (zz >= -LakeShoreEPS)) goto
nocalc300;
2242
2243             x33 = x11 + zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * cos(fai77) / n12;
2244             y33 = y11 - zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * sin(fai77) / n12;
2245
2246             Rtemp = 0.0;
2247             Gtemp = 0.0;
2248             Btemp = 0.0;
2249
2250
2251             Hitpath0 = true;
2252             scale1 = 1.0;

```

```

2253
2254     Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
2255     Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
2256     if (1.0 - 1.0 / (n12 * n12) * pow(cos(theta77), 2) < 0.0) Hitpath0 =
        false;
2257     else Resultvector1[3] = -sqrt(1.0 - 1.0 / (n12 * n12) * pow(cos
        (theta77), 2));
2258
2259     if (Hitpath0 == false) goto nocalc300;
2260
2261     b11x = 0.0;
2262     b11y = 0.0;
2263     b11z = 0.0;
2264
2265     Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, zz, xobs, yobs,
        zobs, n12);
2266
2267     if (Hitpath0 == false) goto nocalc300;
2268
2269     if (b11z == 0)
2270     {
2271         Hitpath0 = false;
2272         goto nocalc300;
2273     }
2274
2275     s2 = (0.0 - zz) / b11z;
2276     x4888 = x11 + s2 * b11x;
2277     y4888 = y11 + s2 * b11y;
2278
2279     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
2280     reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin
        (theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan
        (theta77 + thetadash7), 2)));
2281     beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(
        y4888 - yobs, 2)));
2282     alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
        y4888, 2)));
2283     reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin
        (beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
2284
2285     CosScatAng = Resultvector1[1] * b11x + Resultvector1[2] * b11y +
        Resultvector1[3] * b11z;
2286     ScatAng = acos(CosScatAng); // in the unit of rad
2287
2288     L3 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(zz, 2));
2289     L4 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(zz,
        2));
2290
2291     dS = Xgrid * Ygrid;
2292
2293     Xsum = 0.0;
2294     Ysum = 0.0;
2295     Zsum = 0.0;
2296
2297     bS = 0.0;
2298
2299     for (n = 1; n <= 70; n++)
2300     {
2301         wavel = 380.0 + (static_cast<double>(n - 1)) * deltaLamda;
2302         Exp1 = exp(-L3 * (k1 + bw[n] + Absorbance[n] + MieScatC_array[n]
            + Miec * 650.0 / wavel));
2303
2304         for (i2 = 1; i2 <= 2 * nang - 1; i2++)
2305             s11[i2] = s11_table[n][i2];

```

```

2306
2307         if (SumPhaseFunc_array[n] > 0.0)
2308         {
2309             PhaseFunc = BHMIE1_fromTable(ScatAng, s11, nang) /
                SumPhaseFunc_array[n];
2310         }
2311         else
2312         {
2313             PhaseFunc = 0.0;
2314         }
2315         Lrad = Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
            reffactor22 * Exp1 * PhaseFunc * MieScatC_array[n] * dS / (n12
            * n12) + Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
            reffactor22 * 1.0 * PhaseFunc * MieScatC_array[n] * dS / (
            n12 * n12) * IZ1d[(n - 1) * (int(d / FactorZ + 0.001) + 1) +
            k];
2316         Eobs = Lrad * exp(-L4 * (k1 + bw[n] + Absorbance[n] +
            MieScatC_array[n] + Miec * 650.0 / wavel));
2317         Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
2318         Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
2319         Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
2320     }
2321
2322     R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
2323     G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
2324     B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
2325
2326     if (R_val < 0.0) R_val = 0.0;
2327     else R_val = R_val * FactorDC;
2328
2329     if (G_val < 0.0) G_val = 0.0;
2330     else G_val = G_val * FactorDC;
2331
2332     if (B_val < 0.0) B_val = 0.0;
2333     else B_val = B_val * FactorDC;
2334
2335     if (Hitpath0 == false) goto nocalc300;
2336
2337     for (i1 = 0; i1 <= MaxX * FacDetailed; i1++)
2338     for (j1 = 0; j1 <= MaxY * FacDetailed; j1++)
2339     {
2340         xtemp = (static_cast<double>(i1) - 0.5 * static_cast<double>
            (MaxX * FacDetailed)) / static_cast<double>(MaxX *
            FacDetailed) * Xrange;
2341         ytemp = (static_cast<double>(j1) - 0.5 * static_cast<double>
            (MaxY * FacDetailed)) / static_cast<double>(MaxY *
            FacDetailed) * Yrange;
2342
2343         x111 = xtemp * cos(epsiron) + ytemp * sin(epsiron);
2344         y111 = -xtemp * sin(epsiron) + ytemp * cos(epsiron);
2345
2346         xtemp = x111;
2347         ytemp = y111;
2348
2349
2350         Rtemp = R_val;
2351         if (Rtemp < 0.0) Rtemp = 0.0;
2352         Gtemp = G_val;
2353         if (Gtemp < 0.0) Gtemp = 0.0;
2354         Btemp = B_val;
2355         if (Btemp < 0.0) Btemp = 0.0;
2356
2357         Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow(x4888 -
            xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 * pow(sigma,

```

```

2358         2));
2359         R_Mie_2d[i1][j1] = R_Mie_2d[i1][j1] + Rtemp * Gauss;
2360         G_Mie_2d[i1][j1] = G_Mie_2d[i1][j1] + Gtemp * Gauss;
2361         B_Mie_2d[i1][j1] = B_Mie_2d[i1][j1] + Btemp * Gauss;
2362     }
2363     nocalc300:
2364     ;
2365     }
2366     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2367     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2368     {
2369         R_Mie_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =
2370             R_Mie_2d[i1][j1];
2371         G_Mie_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =
2372             G_Mie_2d[i1][j1];
2373         B_Mie_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =
2374             B_Mie_2d[i1][j1];
2375     }
2376     return 1;
2377 }
2378 double BHMIE1_fromTable(double Angle, double* s11, int nang)
2379 {
2380     double Pi;
2381     Pi = 3.14159265358979;
2382
2383     double dang;
2384
2385     dang = Pi / 2.0 / (static_cast<double>(nang - 1));
2386
2387     int i;
2388     double AngStart, AngEnd;
2389
2390     double s11_val;
2391
2392     for (i = 1; i <= 2 * nang - 1; i++)
2393     {
2394         if (i == 1) AngStart = 0.1 / 180.0 * Pi;
2395         else AngStart = (static_cast<double>(i - 1)) * dang;
2396
2397         AngEnd = (static_cast<double>(i)) * dang;
2398
2399         if ((AngStart <= Angle) && (Angle < AngEnd))
2400         {
2401             s11_val = s11[i] + (Angle - AngStart) / (AngEnd - AngStart) * (s11[i +
2402             1] - s11[i]);
2403             return s11_val;
2404         }
2405     }
2406     return 0.0;
2407 }
2408 }
2409
2410
2411
2412
2413 int MieSca_Rect_2nd(double* Iz1d, double* R_Mie_2d_1, double* G_Mie_2d_1, double*
2414     B_Mie_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int MaxX,
2415     int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double MieC,
2416     double* MieScatC_array)

```



```
2414 {
2415     int i, j;
2416     double x11, y11, z11;
2417     double Rtemp, Gtemp, Btemp;
2418     bool Hitpath0;
2419     double scale1;
2420     double Resultvector1[5];
2421     double b11x, b11y, b11z;
2422     double x33, y33;
2423     double s2;
2424     double x4888, y4888;
2425     double thetadash7;
2426     double reffactor22;
2427     double Pi;
2428     double alfa, beta;
2429     double Xsum, Ysum, Zsum;
2430     int n;
2431     double Exp1;
2432     double R_val, G_val, B_val;
2433     long i1, j1;
2434     double xtemp, ytemp;
2435     double Gauss;
2436     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,
        FacDetailed, LakeShoreEPS;
2437     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,
        Rittaikaku, refl;
2438     int k;
2439     double zz;
2440     double FactorZ;
2441     double CosScatAng, delta0, PhaseFunc, L3, L4;
2442     double ScatAng;
2443     double b11x1, b11y1, b11z1;
2444     double FlagStepAgainstSlope, InitZ, theta1;
2445     double Xrange, Yrange;
2446
2447     double Lrad, Eobs;
2448
2449     double dummy1;
2450
2451     double iPetzoldPF;
2452     double a6, a5, a4, a3, a2, a1, a0;
2453
2454     double y0;
2455
2456     FactorDC = Params1[0];
2457     gzai = Params1[1];
2458     sigma = Params1[2];
2459     delta = Params1[3];
2460     deltaLamda = Params1[4];
2461     r1 = Params1[5];
2462     d = Params1[6];
2463     Xgrid = Params1[7];
2464     Ygrid = Params1[8];
2465     FacDetailed = Params1[9];
2466     LakeShoreEPS = Params1[10];
2467     fai77 = Params1[11];
2468     theta77 = Params1[12];
2469     n12 = Params1[13];
2470     xobs = Params1[14];
2471     yobs = Params1[15];
2472     zobs = Params1[16];
2473     k1 = Params1[17];
2474     Km = Params1[18];
2475     SettleShodo = Params1[19];
```

```

2476 TheoShodo = Params1[20];
2477 Rittaikaku = Params1[21];
2478 refl = Params1[22];
2479 delta0 = Params1[23];
2480 FactorZ = Params1[24];
2481 FlagStepAgainstSlope = Params1[25];
2482 InitZ = Params1[26];
2483 theta1 = Params1[27];
2484 iPetzoldPF = Params1[28];
2485 y0 = Params1[29];
2486
2487 a6 = 0.1571;
2488 a5 = -0.4145;
2489 a4 = -0.0804;
2490 a3 = 0.6345;
2491 a2 = -0.2644;
2492 a1 = -1.7688;
2493 a0 = 1.8269;
2494
2495 Xrange = MaxX * Xgrid;
2496 Yrange = MaxY * Ygrid;
2497
2498 double** R_Mie_2d;
2499 double** G_Mie_2d;
2500 double** B_Mie_2d;
2501
2502 double dS;
2503
2504 double epsilon;
2505 epsilon = atan2(xobs, yobs);
2506 double x111, y111;
2507
2508 double wl;
2509
2510 R_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2511 G_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2512 B_Mie_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2513
2514 for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2515     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2516     {
2517         R_Mie_2d[i1][j1] = 0.0;
2518         G_Mie_2d[i1][j1] = 0.0;
2519         B_Mie_2d[i1][j1] = 0.0;
2520     }
2521
2522 for (i1 = 0; i1 <= (MaxX * static_cast<int>(FacDetailed) + 0) * (MaxY *
static_cast<int>(FacDetailed) + 0); i1++)
2523 {
2524     R_Mie_2d_1[i1] = 0.0;
2525     G_Mie_2d_1[i1] = 0.0;
2526     B_Mie_2d_1[i1] = 0.0;
2527 }
2528
2529 Pi = 3.14159265358979;
2530
2531 for (i = 0; i <= MaxX; i++)
2532     for (j = 0; j <= MaxY; j++)
2533         for (k = 0; k <= Maxk; k++)
2534         {
2535             x11 = (static_cast<double>(i) - 0.5 * static_cast<double>(MaxX)) /

```

```

static_cast<double>(MaxX) * Xrange;
2536 y11 = (static_cast<double>(j) - 0.5 * static_cast<double>(MaxY)) /
static_cast<double>(MaxY) * Yrange;
2537
2538 x111 = x11 * cos(epsiron) + y11 * sin(epsiron);
2539 y111 = -x11 * sin(epsiron) + y11 * cos(epsiron);
2540
2541 x11 = x111;
2542 y11 = y111;
2543
2544 if (FlagStepAgainstSlope == 1.0)
2545 {
2546     z11 = -d;
2547     theta1 = 0.0;
2548 }
2549 else
2550 {
2551     z11 = -(static_cast<double>(j) / static_cast<double>(MaxY) *
Yrange) * tan(theta1) - InitZ;
2552     d = InitZ + Yrange * tan(theta1);
2553 }
2554
2555 zz = -double(k) * FactorZ;
2556
2557 if ((z11 - zz > -LakeShoreEPS) || (zz >= -LakeShoreEPS)) goto
nocalc300;
2558
2559 x33 = x11 + zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * cos(fai77) / n12;
2560 y33 = y11 - zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * sin(fai77) / n12;
2561
2562 Rtemp = 0.0;
2563 Gtemp = 0.0;
2564 Btemp = 0.0;
2565
2566 Hitpath0 = true;
2567 scale1 = 1.0;
2568
2569 Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
2570 Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
2571 if (1.0 - 1.0 / (n12 * n12) * pow(cos(theta77), 2) < 0.0) Hitpath0 =
false;
2572 else Resultvector1[3] = -sqrt(1.0 - 1.0 / (n12 * n12) * pow(cos
(theta77), 2));
2573
2574 if (Hitpath0 == false) goto nocalc300;
2575
2576 b11x = 0.0;
2577 b11y = 0.0;
2578 b11z = 0.0;
2579
2580 Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, zz, xobs, yobs,
zobs, n12);
2581
2582 b11x1 = b11x;
2583 b11y1 = b11y;
2584 b11z1 = b11z;
2585
2586 if (Hitpath0 == false) goto nocalc300;
2587
2588 if (b11z == 0.0)

```

```

2591     {
2592         Hitpath0 = false;
2593         goto nocalc300;
2594     }
2595
2596     s2 = (0.0 - zz) / b11z;
2597     x4888 = x11 + s2 * b11x;
2598     y4888 = y11 + s2 * b11y;
2599
2600     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
2601     reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin
        (theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan
        (theta77 + thetadash7), 2)));
2602     beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(
        y4888 - yobs, 2)));
2603     alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
        y4888, 2)));
2604     reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin
        (beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
2605
2606     CosScatAng = Resultvector1[1] * b11x + Resultvector1[2] * b11y +
        Resultvector1[3] * b11z;
2607     ScatAng = acos(CosScatAng) / Pi * 180;
2608
2609     if (fabs(iPetzoldPF - 1) < 0.01)
2610     {
2611         dummy1 = a6 * pow(log10(ScatAng), 6) + a5 * pow(log10(ScatAng),
            5) + a4 * pow(log10(ScatAng), 4) + a3 * pow(log10(ScatAng), 3)
            + a2 * pow(log10(ScatAng), 2) + a1 * log10(ScatAng) + a0;
2612         PhaseFunc = pow(10, dummy1);
2613     }
2614     else
2615     {
2616         PhaseFunc = 0.0;
2617     }
2618
2619     L3 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(zz, 2));
2620     L4 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(zz,
        2));
2621
2622     dS = Xgrid * Ygrid;
2623
2624     Xsum = 0;
2625     Ysum = 0;
2626     Zsum = 0;
2627
2628     for (n = 1; n < 70; n++)
2629     {
2630         wl = 380.0 + static_cast<double>((n - 1) * 5);
2631         Exp1 = exp(-L3 * (k1 + bw[n] + Absorbance[n] + MieScatC_array[n]
            + Miec * 650.0 / wl));
2632         Lrad = Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
            reffactor22 * Exp1 * PhaseFunc * Miec * 650.0 / wl * dS / (n12
            * n12) + Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
            reffactor22 * 1.0 * PhaseFunc * Miec * 650.0 / wl * dS / (
            n12 * n12) * lz1d[(n - 1) * (int(d / FactorZ + 0.001) + 1) +
            k];
2633         Eobs = Lrad * exp(-L4 * (k1 + bw[n] + Absorbance[n] +
            MieScatC_array[n] + Miec * 650.0 / wl));
2634         Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
2635         Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
2636         Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
2637     }
2638

```

```

2639     R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
2640     G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
2641     B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
2642
2643     if (R_val < 0.0) R_val = 0.0;
2644     else R_val = R_val * FactorDC;
2645
2646     if (G_val < 0.0) G_val = 0.0;
2647     else G_val = G_val * FactorDC;
2648
2649     if (B_val < 0.0) B_val = 0.0;
2650     else B_val = B_val * FactorDC;
2651
2652     if (Hitpath0 == false) goto nocalc300;
2653
2654     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2655     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2656     {
2657         xtemp = (static_cast<double>(i1) - 0.5 * static_cast<double>(
                (MaxX * FacDetailed)) / static_cast<double>(MaxX *
                FacDetailed) * Xrange;
2658         ytemp = (static_cast<double>(j1) - 0.5 * static_cast<double>(
                (MaxY * FacDetailed)) / static_cast<double>(MaxY *
                FacDetailed) * Yrange;
2659
2660         x111 = xtemp * cos(epsiron) + ytemp * sin(epsiron);
2661         y111 = -xtemp * sin(epsiron) + ytemp * cos(epsiron);
2662
2663         xtemp = x111;
2664         ytemp = y111;
2665
2666
2667         Rtemp = R_val;
2668         if (Rtemp < 0.0) Rtemp = 0.0;
2669         Gtemp = G_val;
2670         if (Gtemp < 0.0) Gtemp = 0.0;
2671         Btemp = B_val;
2672         if (Btemp < 0.0) Btemp = 0.0;
2673
2674         Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow(x4888 -
                xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 * pow(sigma,
                2)));
2675
2676         R_Mie_2d[i1][j1] = R_Mie_2d[i1][j1] + Rtemp * Gauss;
2677         G_Mie_2d[i1][j1] = G_Mie_2d[i1][j1] + Gtemp * Gauss;
2678         B_Mie_2d[i1][j1] = B_Mie_2d[i1][j1] + Btemp * Gauss;
2679     }
2680     nocalc300:
2681     ;
2682 }
2683
2684     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2685     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2686     {
2687         R_Mie_2d_1[i1 * (MaxY * int(FacDetailed) + 1) + j1] = R_Mie_2d[i1][j1];
2688         G_Mie_2d_1[i1 * (MaxY * int(FacDetailed) + 1) + j1] = G_Mie_2d[i1][j1];
2689         B_Mie_2d_1[i1 * (MaxY * int(FacDetailed) + 1) + j1] = B_Mie_2d[i1][j1];
2690     }
2691
2692     return 1;
2693 }
2694 }
2695
2696

```

```
2697 int DenstFlucSca_Rect_2nd(double* Izid, double* R_ES_2d_1, double* G_ES_2d_1,
    double* B_ES_2d_1, double* Params1, double* Xbar, double* Ybar, double* Zbar, int
    MaxX, int MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double
    MieC, double* MieScatC_array)
2698 {
2699     int i, j;
2700     double x11, y11, z11;
2701     double Rtemp, Gtemp, Btemp;
2702     bool Hitpath0;
2703     double scale1;
2704     double b11x, b11y, b11z;
2705     double x33, y33;
2706     double s2;
2707     double x4888, y4888;
2708     double thetadash7;
2709     double reffactor22;
2710     double Pi;
2711     double alfa, beta;
2712     double Xsum, Ysum, Zsum;
2713     int n;
2714     double Exp1;
2715     double R_val, G_val, B_val;
2716     long i1, j1;
2717     double xtemp, ytemp;
2718     double Gauss;
2719     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,
        FacDetailed, LakeShoreEPS;
2720     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,
        Rittaikaku, refl;
2721     int k;
2722     double zz;
2723     double FactorZ;
2724     double CosScatAng, beta90factor, delta0, PhaseFunc, L3, L4;
2725     double b11x1, b11y1, b11z1;
2726     double FlagStepAgainstSlope, InitZ, theta1;
2727     double Xrange, Yrange;
2728
2729     double Lrad, Eobs;
2730
2731     double Resultvector1[5];
2732
2733     double y0;
2734
2735     FactorDC = Params1[0];
2736     gzai = Params1[1];
2737     sigma = Params1[2];
2738     delta = Params1[3];
2739     deltaLamda = Params1[4];
2740     r1 = Params1[5];
2741     d = Params1[6];
2742     Xgrid = Params1[7];
2743     Ygrid = Params1[8];
2744     FacDetailed = Params1[9];
2745     LakeShoreEPS = Params1[10];
2746     fai77 = Params1[11];
2747     theta77 = Params1[12];
2748     n12 = Params1[13];
2749     xobs = Params1[14];
2750     yobs = Params1[15];
2751     zobs = Params1[16];
2752     k1 = Params1[17];
2753     Km = Params1[18];
2754     SettleShodo = Params1[19];
2755     TheoShodo = Params1[20];
```

```

2756 Rittaikaku = Params1[21];
2757 refl = Params1[22];
2758 delta0 = Params1[23];
2759 FactorZ = Params1[24];
2760 FlagStepAgainstSlope = Params1[25];
2761 InitZ = Params1[26];
2762 theta1 = Params1[27];
2763 y0 = Params1[28];
2764
2765 Xrange = MaxX * Xgrid;
2766 Yrange = MaxY * Ygrid;
2767
2768
2769 double** R_ES_2d;
2770 double** G_ES_2d;
2771 double** B_ES_2d;
2772
2773 double dS;
2774
2775 double epsiron;
2776 epsiron = atan2(xobs, yobs);
2777 double x111, y111;
2778
2779 double wl;
2780
2781 R_ES_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2782 G_ES_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2783 B_ES_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
static_cast<int>(FacDetailed) + 1);
2784
2785 for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2786     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2787     {
2788         R_ES_2d[i1][j1] = 0.0;
2789         G_ES_2d[i1][j1] = 0.0;
2790         B_ES_2d[i1][j1] = 0.0;
2791     }
2792
2793 for (i1 = 0; i1 <= (MaxX * static_cast<int>(FacDetailed) + 0) * (MaxY *
static_cast<int>(FacDetailed) + 0); i1++)
2794 {
2795     R_ES_2d_1[i1] = 0.0;
2796     G_ES_2d_1[i1] = 0.0;
2797     B_ES_2d_1[i1] = 0.0;
2798 }
2799
2800 Pi = 3.14159265358979;
2801
2802 for (i = 0; i <= MaxX; i++)
2803     for (j = 0; j <= MaxY; j++)
2804         for (k = 0; k <= Maxk; k++)
2805         {
2806             x11 = (static_cast<double>(i) - 0.5 * static_cast<double>(MaxX)) /
static_cast<double>(MaxX) * Xrange;
2807             y11 = (static_cast<double>(j) - 0.5 * static_cast<double>(MaxY)) /
static_cast<double>(MaxY) * Yrange;
2808
2809             x111 = x11 * cos(epsiron) + y11 * sin(epsiron);
2810             y111 = -x11 * sin(epsiron) + y11 * cos(epsiron);
2811
2812             x11 = x111;
2813             y11 = y111;

```

```

2814
2815         if (FlagStepAgainstSlope == 1.0)
2816         {
2817             z11 = -d;
2818             theta1 = 0.0;
2819         }
2820     else
2821     {
2822         z11 = -(static_cast<double>(j) / static_cast<double>(MaxY) *
2823             Yrange) * tan(theta1) - InitZ;
2824         d = InitZ + Yrange * tan(theta1);
2825     }
2826
2827     zz = -double(k) * FactorZ;
2828
2829     if ((z11 - zz > -LakeShoreEPS) || (zz >= -LakeShoreEPS)) goto
2830         nocalc200;
2831
2832     Rtemp = 0.0;
2833     Gtemp = 0.0;
2834     Btemp = 0.0;
2835
2836     Hitpath0 = true;
2837     scale1 = 1.0;
2838
2839     Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
2840     Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
2841     if (1.0 - 1.0 / (n12 * n12) * pow(cos(theta77), 2) < 0.0) Hitpath0 =
2842         false;
2843     else Resultvector1[3] = -sqrt(1.0 - 1.0 / (n12 * n12) * pow(cos
2844         (theta77), 2));
2845
2846     if (Hitpath0 == false) goto nocalc200;
2847
2848     x33 = x11 + zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
2849         (theta77) * cos(fai77) / n12;
2850     y33 = y11 - zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
2851         (theta77) * sin(fai77) / n12;
2852
2853     b11x = 0.0;
2854     b11y = 0.0;
2855     b11z = 0.0;
2856
2857     Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, zz, xobs, yobs,
2858         zobs, n12);
2859
2860     if (Hitpath0 == false) goto nocalc200;
2861
2862     b11x1 = b11x;
2863     b11y1 = b11y;
2864     b11z1 = b11z;
2865
2866     if (b11z == 0.0)
2867     {
2868         Hitpath0 = false;
2869         goto nocalc200;
2870     }
2871
2872     s2 = (0.0 - zz) / b11z;
2873     x4888 = x11 + s2 * b11x;
2874     y4888 = y11 + s2 * b11y;
2875
2876     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));

```



```

2871 reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin
      (theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan
      (theta77 + thetadash7), 2)));
2872 beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(
      y4888 - yobs, 2)));
2873 alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
      y4888, 2)));
2874 reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin
      (beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
2875
2876 CosScatAng = Resultvector1[1] * b11x + Resultvector1[2] * b11y +
      Resultvector1[3] * b11z;
2877 beta90factor = 1.0 / (8.0 * Pi / 3.0 * (2.0 + delta0) / (1.0 +
      delta0));
2878 PhaseFunc = beta90factor * (1.0 + (1.0 - delta0) / (1.0 + delta0) *
      pow(CosScatAng, 2));
2879
2880 L3 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(zz, 2));
2881 L4 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(zz,
      2));
2882
2883 dS = Xgrid * Ygrid;
2884
2885 Xsum = 0.0;
2886 Ysum = 0.0;
2887 Zsum = 0.0;
2888
2889 for (n = 1; n < 70; n++)
2890 {
2891     wl = 380.0 + static_cast<double>((n - 1) * 5);
2892
2893     Exp1 = exp(-L3 * (k1 + bw[n] + Absorbance[n] + MieScatC_array[n]
      + MieC * 650.0 / wl));
2894     Lrad = Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
      reffactor22 * Exp1 * PhaseFunc * bw[n] * dS / (n12 * n12) + Km
      * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
      reffactor22 * 1.0 * PhaseFunc * bw[n] * dS / (n12 * n12) * lz1d
      [(n - 1) * (int)(d / FactorZ + 0.001) + 1) + k];
2895     Eobs = Lrad * exp(-L4 * (k1 + bw[n] + Absorbance[n] +
      MieScatC_array[n] + MieC * 650.0 / wl));
2896     Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
2897     Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
2898     Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
2899 }
2900
2901 R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
2902 G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
2903 B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
2904
2905 if (R_val < 0.0) R_val = 0.0;
2906 else R_val = R_val * FactorDC;
2907
2908 if (G_val < 0.0) G_val = 0.0;
2909 else G_val = G_val * FactorDC;
2910
2911 if (B_val < 0.0) B_val = 0.0;
2912 else B_val = B_val * FactorDC;
2913
2914 if (Hitpath0 == false) goto nocalc200;
2915
2916 for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2917     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2918     {
2919         xtemp = (static_cast<double>(i1) - 0.5 * static_cast<double>

```

```

        (MaxX * FacDetailed)) / static_cast<double>(MaxX *
        FacDetailed) * Xrange;
2920 ytemp = (static_cast<double>(j1) - 0.5 * static_cast<double>
        (MaxY * FacDetailed)) / static_cast<double>(MaxY *
        FacDetailed) * Yrange;
2921
2922 x111 = xtemp * cos(epsiron) + ytemp * sin(epsiron);
2923 y111 = -xtemp * sin(epsiron) + ytemp * cos(epsiron);
2924
2925 xtemp = x111;
2926 ytemp = y111;
2927
2928 Rtemp = R_val;
2929 if (Rtemp < 0.0) Rtemp = 0.0;
2930 Gtemp = G_val;
2931 if (Gtemp < 0.0) Gtemp = 0.0;
2932 Btemp = B_val;
2933 if (Btemp < 0.0) Btemp = 0.0;
2934
2935 Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow(x4888 -
        xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 * pow(sigma,
        2)));
2936
2937 R_ES_2d[i1][j1] = R_ES_2d[i1][j1] + Rtemp * Gauss;
2938 G_ES_2d[i1][j1] = G_ES_2d[i1][j1] + Gtemp * Gauss;
2939 B_ES_2d[i1][j1] = B_ES_2d[i1][j1] + Btemp * Gauss;
2940
2941     }
2942     nocalc200:
2943     ;
2944     }
2945     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
2946     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
2947     {
2948         R_ES_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] = R_ES_2d
        [i1][j1];
2949         G_ES_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] = G_ES_2d
        [i1][j1];
2950         B_ES_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] = B_ES_2d
        [i1][j1];
2951     }
2952
2953     return 1;
2954 }
2955
2956
2957
2958 int IrrRefBottom_Rect_2nd(double* Lz1d, double* R_Bot_2d_1, double* G_Bot_2d_1,
        double* B_Bot_2d_1, double* Params, double* Xbar, double* Ybar, double* Zbar, int
        MaxX, int MaxY, double* bw, double* Absorbance, double* f_fine, double MieC,
        double* MieScatC_array)
2959 {
2960     int i, j;
2961     double x11, y11, z11;
2962     double Rtemp, Gtemp, Btemp;
2963     bool Hitpath0;
2964     double scale1;
2965     double b11x, b11y, b11z;
2966     double x33, y33;
2967     double s2;
2968     double x4888, y4888;
2969     double L1, L2;
2970     double thetadash7;
2971     double reffactor22;

```

```
2972 double Pi;
2973 double alfa, beta;
2974 double Xsum, Ysum, Zsum;
2975 int n;
2976 double Exp1;
2977 double R_val, G_val, B_val;
2978 long i1, j1;
2979 double xtemp, ytemp;
2980 double Gauss;
2981 double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,
    FacDetailed, LakeShoreEPS;
2982 double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,
    Rittaikaku, refl;
2983 double FlagStepAgainstSlope, InitZ, theta1;
2984 double Xrange, Yrange;
2985
2986 double Lrad, Eobs;
2987
2988 double y0;
2989
2990 double iLambertian;
2991 double FactorZ;
2992
2993 double dS;
2994
2995
2996 FactorDC = Params[0];
2997 gzai = Params[1];
2998 sigma = Params[2];
2999 delta = Params[3];
3000 deltaLamda = Params[4];
3001 r1 = Params[5];
3002 d = Params[6];
3003 Xgrid = Params[7];
3004 Ygrid = Params[8];
3005 FacDetailed = Params[9];
3006 LakeShoreEPS = Params[10];
3007 fai77 = Params[11];
3008 theta77 = Params[12];
3009 n12 = Params[13];
3010 xobs = Params[14];
3011 yobs = Params[15];
3012 zobs = Params[16];
3013 k1 = Params[17];
3014 Km = Params[18];
3015 SettleShodo = Params[19];
3016 TheoShodo = Params[20];
3017 Rittaikaku = Params[21];
3018 refl = Params[22];
3019 FlagStepAgainstSlope = Params[23];
3020 InitZ = Params[24];
3021 theta1 = Params[25];
3022 y0 = Params[26];
3023 iLambertian = Params[27];
3024 FactorZ = Params[28];
3025
3026 double epsiron;
3027 epsiron = atan2(xobs, yobs);
3028 double x111, y111;
3029
3030 Xrange = MaxX * Xgrid;
3031 Yrange = MaxY * Ygrid;
3032
3033 double** R_Bot_2d;
```

```

3034     double** G_Bot_2d;
3035     double** B_Bot_2d;
3036
3037     int NumBot;
3038
3039     double wI;
3040
3041
3042     R_Bot_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
3043         static_cast<int>(FacDetailed) + 1);
3044     G_Bot_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
3045         static_cast<int>(FacDetailed) + 1);
3046     B_Bot_2d = alloc_matrix(MaxX * static_cast<int>(FacDetailed) + 1, MaxY *
3047         static_cast<int>(FacDetailed) + 1);
3048
3049     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
3050     {
3051         for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
3052         {
3053             R_Bot_2d[i1][j1] = 0.0;
3054             G_Bot_2d[i1][j1] = 0.0;
3055             B_Bot_2d[i1][j1] = 0.0;
3056         }
3057     }
3058
3059     for (i1 = 0; i1 <= (MaxX * int(FacDetailed) + 0) * (MaxY * int(FacDetailed) +
3060         0); i1++)
3061     {
3062         R_Bot_2d_1[i1] = 0.0;
3063         G_Bot_2d_1[i1] = 0.0;
3064         B_Bot_2d_1[i1] = 0.0;
3065     }
3066
3067     Pi = 3.14159265358979;
3068
3069     for (i = 0; i <= MaxX; i++)
3070     {
3071         for (j = 0; j <= MaxY; j++)
3072         {
3073             x11 = (static_cast<double>(i) - 0.5 * static_cast<double>(MaxX)) /
3074                 static_cast<double>(MaxX) * Xrange;
3075             y11 = (static_cast<double>(j) - 0.5 * static_cast<double>(MaxY)) /
3076                 static_cast<double>(MaxY) * Yrange;
3077
3078             x111 = x11 * cos(epsiron) + y11 * sin(epsiron);
3079             y111 = -x11 * sin(epsiron) + y11 * cos(epsiron);
3080
3081             x11 = x111;
3082             y11 = y111;
3083
3084             if (fabs(FlagStepAgainstSlope - 1.0) < 0.01)
3085             {
3086                 z11 = -d;
3087                 theta1 = 0.0;
3088             }
3089             else
3090             {
3091                 z11 = -(static_cast<double>(j) / static_cast<double>(MaxY) * Yrange)
3092                     * tan(theta1) - InitZ;
3093                 d = InitZ + Yrange * tan(theta1);
3094             }
3095
3096             Rtemp = 0.0;
3097             Gtemp = 0.0;
3098             Btemp = 0.0;

```

```

3091
3092     Hitpath0 = true;
3093     scale1 = 1.0;
3094
3095     if (Hitpath0 == false) goto nocalc000;
3096
3097     x33 = x11 + z11 / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos(
3098         theta77) * cos(fai77) / n12;
3099     y33 = y11 - z11 / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos(
3100         theta77) * sin(fai77) / n12;
3101
3102     b11x = 0.0;
3103     b11y = 0.0;
3104     b11z = 0.0;
3105
3106     Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, z11, xobs, yobs,
3107         zobs, n12);
3108
3109     if (b11z == 0.0)
3110     {
3111         Hitpath0 = false;
3112         goto nocalc000;
3113     }
3114
3115     if (Hitpath0 == false) goto nocalc000;
3116
3117     s2 = (0.0 - z11) / b11z;
3118     x4888 = x11 + s2 * b11x;
3119     y4888 = y11 + s2 * b11y;
3120
3121     L1 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(z11, 2));
3122     L2 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(z11, 2));
3123
3124     if (Hitpath0 == false) goto nocalc000;
3125
3126     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
3127     reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin(
3128         theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan(
3129         theta77 + thetadash7), 2)));
3130
3131     beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(y4888 -
3132         yobs, 2)));
3133     alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
3134         y4888, 2)));
3135     reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin(
3136         beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
3137
3138     if (Hitpath0 == false) goto nocalc000;
3139
3140     dS = Xgrid * Ygrid;
3141
3142     Xsum = 0.0;
3143     Ysum = 0.0;
3144     Zsum = 0.0;
3145
3146     NumBot = int(-z11 / FactorZ + 0.001);
3147
3148     for (n = 1; n <= 70; n++)
3149     {
3150         wl = 380.0 + static_cast<double>((n - 1) * 5);
3151
3152         Exp1 = exp(-L1 * (k1 + bw[n] + Absorbance[n] + MieScatC_array[n] +
3153             MieC * 650.0 / wl));
3154         if (fabs(iLambertian - 1) < 0.01)
3155         {

```

```

3146      Lrad = Km * f_fine[n] * SettleShodo / TheoShodo * reffactor22 *
      Exp1 * refl / Pi * dS / (n12 * n12) + Km * f_fine[n] *
      SettleShodo / TheoShodo * reffactor22 * 1.0 * refl / Pi * dS /
      (n12 * n12) * Lz1d[(n - 1) * (NumBot + 1) + NumBot + 1 - 1];
3147      Eobs = Lrad * b11z * exp(-L2 * (k1 + bw[n] + Absorbance[n] +
      MieScatC_array[n] + MieC * 650.0 / wl));
3148  }
3149  else
3150  {
3151      Eobs = 0.0;
3152  }
3153
3154      Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
3155      Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
3156      Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
3157  }
3158
3159      R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
3160      G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
3161      B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
3162
3163      if (R_val < 0.0) R_val = 0.0;
3164      else R_val = R_val * FactorDC;
3165
3166      if (G_val < 0.0) G_val = 0.0;
3167      else G_val = G_val * FactorDC;
3168
3169      if (B_val < 0.0) B_val = 0.0;
3170      else B_val = B_val * FactorDC;
3171
3172      if (Hitpath0 == false) goto nocalc000;
3173
3174      for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
3175      for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
3176      {
3177          xtemp = (static_cast<double>(i1) - 0.5 * static_cast<double>(
      MaxX * FacDetailed)) / static_cast<double>(MaxX * FacDetailed)
      * Xrange;
3178          ytemp = (static_cast<double>(j1) - 0.5 * static_cast<double>(
      MaxY * FacDetailed)) / static_cast<double>(MaxY * FacDetailed)
      * Yrange;
3179
3180          x111 = xtemp * cos(epsiron) + ytemp * sin(epsiron);
3181          y111 = -xtemp * sin(epsiron) + ytemp * cos(epsiron);
3182
3183          xtemp = x111;
3184          ytemp = y111;
3185
3186
3187          Rtemp = R_val;
3188          if (Rtemp < 0.0) Rtemp = 0.0;
3189          Gtemp = G_val;
3190          if (Gtemp < 0.0) Gtemp = 0.0;
3191          Btemp = B_val;
3192          if (Btemp < 0.0) Btemp = 0.0;
3193
3194          Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow(x4888 -
      xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 * pow(sigma, 2)));
3195
3196          R_Bot_2d[i1][j1] = R_Bot_2d[i1][j1] + Rtemp * Gauss;
3197          G_Bot_2d[i1][j1] = G_Bot_2d[i1][j1] + Gtemp * Gauss;
3198          B_Bot_2d[i1][j1] = B_Bot_2d[i1][j1] + Btemp * Gauss;
3199      }
3200

```

```

3201         nocalc000:
3202         ;
3203     }
3204
3205     for (i1 = 0; i1 <= MaxX * static_cast<int>(FacDetailed); i1++)
3206     for (j1 = 0; j1 <= MaxY * static_cast<int>(FacDetailed); j1++)
3207     {
3208         R_Bot_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =  ↗
            R_Bot_2d[i1][j1];
3209         G_Bot_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =  ↗
            G_Bot_2d[i1][j1];
3210         B_Bot_2d_1[i1 * (MaxY * static_cast<int>(FacDetailed) + 1) + j1] =  ↗
            B_Bot_2d[i1][j1];
3211     }
3212
3213     return 1;
3214 }
3215
3216
3217
3218 int splineDLL64(int ScreenX, int ScreenY, double* nn44_1, int* nn55_1, double  ↗
    smax_thre, long MaxIter)
3219 {
3220     int i, j;
3221     long k;
3222     double minn;
3223     double c1, c2, c3;
3224     double s1, s2, s3;
3225     double s;
3226     double smax;
3227     double Omega, sigma1;
3228     double** nn44;
3229     int** nn55;
3230     long longnum;
3231
3232     nn44 = alloc_matrix(long(ScreenX), long(ScreenY));
3233     nn55 = alloc_boolmatrix(long(ScreenX), long(ScreenY));
3234
3235     for (i = 0; i < ScreenX; i++)
3236     {
3237         for (j = 0; j < ScreenY; j++)
3238         {
3239             nn44[i][j] = 0.0;
3240             nn55[i][j] = 0;
3241         }
3242     }
3243
3244     for (i = 0; i < ScreenX; i++)
3245     {
3246         for (j = 0; j < ScreenY; j++)
3247         {
3248             longnum = long(i) * long(ScreenY) + long(j);
3249             nn44[i][j] = nn44_1[longnum];
3250             nn55[i][j] = nn55_1[longnum];
3251         }
3252     }
3253
3254     minn = 0.0;
3255
3256     for (i = 0; i < ScreenX; i++)
3257     for (j = 0; j < ScreenY; j++)
3258     if (nn55[i][j] == 0) nn44[i][j] = minn;
3259
3260     for (i = 0; i < ScreenX; i++)

```

```

3261     for (j = 0; j < ScreenY; j++)
3262         if (nn44[i][j] < 0) nn44[i][j] = 0;
3263
3264     Omega = 0.5;
3265     sigma1 = 5.0;
3266     c1 = (8.0 + sigma1) / (20.0 + 4.0 * sigma1);
3267     c2 = 2.0 / (20.0 + 4.0 * sigma1);
3268     c3 = 0.5 * c2;
3269
3270     for (k = 0; k < MaxIter; k++)
3271     {
3272         for (i = 2; i <= ScreenX - 1 - 3; i++)
3273         {
3274             if (nn55[i][0] == 0) nn44[i][0] = nn44[i][2];
3275             if (nn55[i][1] == 0) nn44[i][1] = nn44[i][2];
3276             if (nn55[i][ScreenY - 1 - 2] == 0) nn44[i][ScreenY - 1 - 2] = nn44[i]  ↗
3277                 [ScreenY - 1 - 3];
3278             if (nn55[i][ScreenY - 1 - 1] == 0) nn44[i][ScreenY - 1 - 1] = nn44[i]  ↗
3279                 [ScreenY - 1 - 3];
3280         }
3281         for (j = 1; j <= ScreenY - 1 - 2; j++)
3282         {
3283             if (nn55[0][j] == 0) nn44[0][j] = nn44[2][j];
3284             if (nn55[1][j] == 0) nn44[1][j] = nn44[2][j];
3285             if (nn55[ScreenX - 1 - 2][j] == 0) nn44[ScreenX - 1 - 2][j] = nn44  ↗
3286                 [ScreenX - 1 - 3][j];
3287             if (nn55[ScreenX - 1 - 1][j] == 0) nn44[ScreenX - 1 - 1][j] = nn44  ↗
3288                 [ScreenX - 1 - 3][j];
3289         }
3290         smax = 0.0;
3291         for (j = 2; j <= ScreenY - 1 - 3; j++)
3292         {
3293             for (i = 2; i <= ScreenX - 1 - 3; i++)
3294             {
3295                 if (nn55[i][j] == 0)
3296                 {
3297                     s1 = nn44[i][j - 1] + nn44[i][j + 1] + nn44[i - 1][j] + nn44[i + 1][j];
3298                     s2 = nn44[i - 1][j - 1] + nn44[i - 1][j + 1] + nn44[i + 1][j - 1] + nn44[i + 1][j + 1];
3299                     s3 = nn44[i - 2][j] + nn44[i + 2][j] + nn44[i][j - 2] + nn44[i][j + 2];
3300                     s = c1 * s1 - c2 * s2 - c3 * s3 - nn44[i][j];
3301                     nn44[i][j] = nn44[i][j] + Omega * s;
3302                     if (fabs(s) > smax) smax = fabs(s);
3303                 }
3304             }
3305         }
3306         if (smax < smax_thre) goto L999;
3307     }
3308
3309     if (k >= MaxIter - 1)
3310     {
3311         for (i = 0; i < ScreenX; i++)
3312         {
3313             for (j = 0; j < ScreenY; j++)
3314             {
3315                 longnum = long(i) * long(ScreenY) + long(j);
3316                 nn44_1[longnum] = nn44[i][j];
3317             }
3318         }
3319         return 0;
3320     }
3321 L999:

```



```

3318     for (i = 0; i < ScreenX; i++)
3319         for (j = 0; j < ScreenY; j++)
3320         {
3321             longnum = long(i) * long(ScreenY) + long(j);
3322             nn44_1[longnum] = nn44[i][j];
3323         }
3324
3325     return 1;
3326 }
3327
3328
3329 int PreSplineLake64(int* T88_1, double** LinHomG, double** LinIHomG, int
    ScreenTrans, int ScreenLong, int ScreenOrigTrans, int ScreenOrigLong, int NumStep,
    double epsiron, double r1, double FacDetailed, double** R_x_2d, double** G_x_2d,
    double** B_x_2d, double* R88_1, double* G88_1, double* B88_1, int*
    ScreenFullIFlag_1, double* MaxYY, double& MinX1, double& MaxX1, double& MinY1,
    double& MaxY1)
3330 {
3331     int i, j;
3332     double x7, y7;
3333     int i1, j1;
3334     double x11, y11;
3335
3336     long LongNum;
3337
3338     double Pi;
3339     Pi = 3.14159265358979;
3340
3341     int response;
3342
3343     int ifactor;
3344
3345     ifactor = 1;
3346
3347     long ScreenNtrans1Long;
3348
3349     ScreenNtrans1Long = static_cast<long>(ScreenTrans) * static_cast<long>
        (ScreenLong);
3350
3351     double** CheckConvertAxisX;
3352     double** CheckConvertAxisY;
3353     double** CheckBeforeConvertAxisX;
3354     double** CheckBeforeConvertAxisY;
3355     double** CheckConvertAxisR;
3356     double** CheckConvertAxisG;
3357     double** CheckConvertAxisB;
3358
3359
3360
3361     CheckConvertAxisX = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3362     CheckConvertAxisY = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3363     CheckBeforeConvertAxisX = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3364     CheckBeforeConvertAxisY = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3365     CheckConvertAxisR = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3366     CheckConvertAxisG = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3367     CheckConvertAxisB = alloc_matrix(static_cast<long>((2.0 * r1 + 1.0) *
        FacDetailed) + 1, static_cast<long>((2.0 * r1 + 1.0) * FacDetailed) + 1);
3368

```

```

3369     bool** Flag8;
3370     bool** Flag8Drawn;
3371     bool** ScreenFlag;
3372     bool** ScreenFullFlag;
3373
3374     double** R8;
3375     double** G8;
3376     double** B8;
3377     double** X8;
3378     double** Y8;
3379     double** ScreenR;
3380     double** ScreenG;
3381     double** ScreenB;
3382     double** ScreenR_Most;
3383     double** ScreenG_Most;
3384     double** ScreenB_Most;
3385     double** R88;
3386     double** G88;
3387     double** B88;
3388
3389     Flag8 = alloc_boolboolmatrix(ScreenTrans, ScreenLong);
3390     Flag8Drawn = alloc_boolboolmatrix(ScreenTrans, ScreenLong);
3391     ScreenFlag = alloc_boolboolmatrix(ScreenTrans, ScreenLong);
3392     ScreenFullFlag = alloc_boolboolmatrix(ScreenTrans * ifactor, ScreenLong *
3393         ifactor);
3394     R8 = alloc_matrix(ScreenTrans, ScreenLong);
3395     G8 = alloc_matrix(ScreenTrans, ScreenLong);
3396     B8 = alloc_matrix(ScreenTrans, ScreenLong);
3397     X8 = alloc_matrix(ScreenTrans, ScreenLong);
3398     Y8 = alloc_matrix(ScreenTrans, ScreenLong);
3399     ScreenR = alloc_matrix(ScreenTrans, ScreenLong);
3400     ScreenG = alloc_matrix(ScreenTrans, ScreenLong);
3401     ScreenB = alloc_matrix(ScreenTrans, ScreenLong);
3402     ScreenR_Most = alloc_matrix(ScreenTrans, ScreenLong);
3403     ScreenG_Most = alloc_matrix(ScreenTrans, ScreenLong);
3404     ScreenB_Most = alloc_matrix(ScreenTrans, ScreenLong);
3405     R88 = alloc_matrix(ScreenTrans, ScreenLong);
3406     G88 = alloc_matrix(ScreenTrans, ScreenLong);
3407     B88 = alloc_matrix(ScreenTrans, ScreenLong);
3408
3409     MaxX1 = 0.0;
3410     MinX1 = double(ScreenTrans) - 1.0;
3411     MaxY1 = 0.0;
3412     MinY1 = double(ScreenLong) - 1.0;
3413
3414     for (i = 0; i < ScreenTrans; i++)    MaxYY[i] = 0.0;
3415
3416     for (i = 0; i <= int(2 * r1 + 1) * FacDetailed; i++)
3417         for (j = 0; j <= int(2 * r1 + 1) * FacDetailed; j++)
3418             {
3419                 CheckConvertAxisX[i][j] = 0.0;
3420                 CheckConvertAxisY[i][j] = 0.0;
3421                 CheckBeforeConvertAxisX[i][j] = 0.0;
3422                 CheckBeforeConvertAxisY[i][j] = 0.0;
3423                 CheckConvertAxisR[i][j] = 0.0;
3424                 CheckConvertAxisG[i][j] = 0.0;
3425                 CheckConvertAxisB[i][j] = 0.0;
3426             }
3427
3428     for (i = 0; i <= int(2 * r1 + 1) * FacDetailed; i++)
3429         for (j = 0; j <= int(2 * r1 + 1) * FacDetailed; j++)
3430             {
3431                 x7 = static_cast<double>(i) / FacDetailed - (r1 + 1.0);

```

```

3432         y7 = static_cast<double>(j) / FacDetailed - (r1 + 1.0);
3433         //XY2XYXY(-epsiron, x7, x11, y7, y11);
3434         x11 = x7;
3435         y11 = y7;
3436
3437         if (x11 * x11 + y11 * y11 < r1 * r1)
3438         {
3439             response=ConvertPhoto2Circle2(ScreenTrans, ScreenLong,
3440                                             ScreenOrigTrans, ScreenOrigLong, false, x11, y11, LinHomG,
3441                                             LinIHomG);
3442
3443             CheckConvertAxisX[i][j] = x11;
3444             CheckConvertAxisY[i][j] = y11;
3445             CheckBeforeConvertAxisX[i][j] = x7;
3446             CheckBeforeConvertAxisY[i][j] = y7;
3447             CheckConvertAxisR[i][j] = R_x_2d[i][j];
3448             CheckConvertAxisG[i][j] = G_x_2d[i][j];
3449             CheckConvertAxisB[i][j] = B_x_2d[i][j];
3450
3451             if (MinX1 > x11) MinX1 = x11;
3452             if (MaxX1 < x11) MaxX1 = x11;
3453             if (MinY1 > y11) MinY1 = y11;
3454             if (MaxY1 < y11) MaxY1 = y11;
3455         }
3456     }
3457
3458     for (i = 0; i < 360; i++)
3459     {
3460         x7 = r1 * cos(static_cast<double>(i) / 180.0 * Pi);
3461         y7 = r1 * sin(static_cast<double>(i) / 180.0 * Pi);
3462         x11 = x7;
3463         y11 = y7;
3464
3465         response = ConvertPhoto2Circle2(ScreenTrans, ScreenLong, ScreenOrigTrans,
3466                                         ScreenOrigLong, false, x11, y11, LinHomG, LinIHomG);
3467
3468         if ((x11 >= 0.0) && (x11 < static_cast<double>(ScreenTrans) - 1.0) && (
3469             y11 >= 0.0) && (y11 < static_cast<double>(ScreenLong) - 1.0)) MaxYY[int
3470             (x11)] = y11;
3471     }
3472
3473     for (i = ScreenTrans - 2; i >= 0; i--)
3474     {
3475         if (MaxYY[i] == 0.0)
3476         {
3477             if (MaxYY[i + 1] > 0.0) MaxYY[i] = MaxYY[i + 1];
3478         }
3479     }
3480
3481     for (i = 1; i <= ScreenTrans - 1; i++)
3482     {
3483         if (MaxYY[i] == 0.0)
3484         {
3485             if (MaxYY[i - 1] > 0.0) MaxYY[i] = MaxYY[i - 1];
3486         }
3487     }
3488
3489     double MaxXX1, MinXX1, MaxYY1, MinYY1;
3490     double x11old;
3491
3492     MaxXX1 = 0.0;
3493     MinXX1 = static_cast<double>(ScreenTrans) - 1.0;
3494     MaxY1 = 0.0;

```

```

3491     for (i1 = 0; i1 <= 360; i1++)
3492     {
3493         x7 = r1 * cos(static_cast<double>(i1) / 180.0 * Pi);
3494         y7 = r1 * sin(static_cast<double>(i1) / 180.0 * Pi);
3495         x11 = x7;
3496         y11 = y7;
3497         response = ConvertPhoto2Circle2(ScreenTrans, ScreenLong, ScreenOrigTrans,
3498             ScreenOrigLong, false, x11, y11, LinHomG, LinIHomG);
3499
3500         if ((x11 >= 0.0) && (x11 < static_cast<double>(ScreenTrans) - 1.0) && (
3501             y11 >= 0.0) && (y11 < static_cast<double>(ScreenLong) - 1.0))
3502         {
3503             if (MaxY1 < y11)
3504             {
3505                 MaxY1 = y11;
3506             }
3507             if (MaxXX1 < x11)
3508             {
3509                 MaxXX1 = x11;
3510                 MaxYY1 = y11;
3511             }
3512             if (MinXX1 > x11)
3513             {
3514                 MinXX1 = x11;
3515                 MinYY1 = y11;
3516             }
3517         }
3518         x11 = r1;
3519         y11 = 0.0;
3520
3521         response = ConvertPhoto2Circle2(ScreenTrans, ScreenLong, ScreenOrigTrans,
3522             ScreenOrigLong, false, x11, y11, LinHomG, LinIHomG);
3523
3524         x11old = x11;
3525         for (i1 = 0; i1 <= 360; i1++)
3526         {
3527             x7 = r1 * cos(static_cast<double>(i1) / 180.0 * Pi);
3528             y7 = r1 * sin(static_cast<double>(i1) / 180.0 * Pi);
3529             x11 = x7;
3530             y11 = y7;
3531             response = ConvertPhoto2Circle2(ScreenTrans, ScreenLong, ScreenOrigTrans,
3532                 ScreenOrigLong, false, x11, y11, LinHomG, LinIHomG);
3533
3534             for (i = 0; i < ScreenTrans; i++)
3535             for (j = 0; j < ScreenLong; j++)
3536             {
3537                 if (((static_cast<double>(j) >= MinYY1) || (static_cast<double>
3538                     (j) >= MaxYY1)) && (MinXX1 <= static_cast<double>(i)) &&
3539                     (static_cast<double>(i) <= MaxXX1) && (static_cast<double>(
3540                         j) > y11) && (x11 <= static_cast<double>(i)) &&
3541                         (static_cast<double>(i) < x11old))
3542                 {
3543                     LongNum = static_cast<long>(i) * static_cast<long>
3544                         (ScreenLong) + static_cast<long>(j);
3545                     T88_1[LongNum] = 0;
3546                 }
3547                 else if (((static_cast<double>(j) <= MinYY1) ||
3548                     (static_cast<double>(j) <= MaxYY1)) && (MinXX1 <=
3549                         static_cast<double>(i)) && (static_cast<double>(i) <= MaxXX1)
3550                     && (static_cast<double>(j) < y11) && (x11old <=
3551                         static_cast<double>(i)) && (static_cast<double>(i) < x11))

```



```

3599         R8[i1][j1] = CheckConvertAxisR[i][j];
3600         G8[i1][j1] = CheckConvertAxisG[i][j];
3601         B8[i1][j1] = CheckConvertAxisB[i][j];
3602     }
3603 }
3604 }
3605 }
3606
3607 for (i = 0; i <= ScreenTrans - 1; i++)
3608     for (j = 0; j <= ScreenLong - 1; j++)
3609         ScreenFlag[i][j] = false;
3610
3611 for (i = 0; i <= ScreenTrans - 1; i = i + ifactor)
3612     for (j = 0; j <= ScreenLong - 1; j = j + ifactor)
3613         for (i1 = 0; i1 <= int(2 * r1 + 1) * FacDetailed; i1++)
3614             for (j1 = 0; j1 <= int(2 * r1 + 1) * FacDetailed; j1++)
3615             {
3616                 if ((static_cast<double>(i) <= CheckConvertAxisX[i1][j1]) &&
3617                     (CheckConvertAxisX[i1][j1] < static_cast<double>(i + ifactor)))
3618                 {
3619                     if ((static_cast<double>(j) <= CheckConvertAxisY[i1][j1]) &&
3620                         (CheckConvertAxisY[i1][j1] < static_cast<double>(j +
3621                             ifactor)))
3622                     {
3623                         if (ScreenFlag[i / ifactor][j / ifactor] == false)
3624                         {
3625                             ScreenFlag[i / ifactor][j / ifactor] = true;
3626                             if (CheckConvertAxisR[i1][j1] >= ScreenR_Most[i /
3627                                 ifactor][j / ifactor])
3628                             {
3629                                 R88[i / ifactor][j / ifactor] = CheckConvertAxisR[i1]
3630                                     [j1];
3631                                 ScreenR_Most[i / ifactor][j / ifactor] = R88[i /
3632                                     ifactor][j / ifactor];
3633                             }
3634                             if (CheckConvertAxisG[i1][j1] >= ScreenG_Most[i /
3635                                 ifactor][j / ifactor])
3636                             {
3637                                 G88[i / ifactor][j / ifactor] = CheckConvertAxisG[i1]
3638                                     [j1];
3639                                 ScreenG_Most[i / ifactor][j / ifactor] = G88[i /
3640                                     ifactor][j / ifactor];
3641                             }
3642                             if (CheckConvertAxisB[i1][j1] >= ScreenB_Most[i /
3643                                 ifactor][j / ifactor])
3644                             {
3645                                 B88[i / ifactor][j / ifactor] = CheckConvertAxisB[i1]
3646                                     [j1];
3647                                 ScreenB_Most[i / ifactor][j / ifactor] = B88[i /
3648                                     ifactor][j / ifactor];
3649                             }
3650                         }
3651                     }
3652                 }
3653             }
3654         }
3655     }
3656
3657 for (i = 0; i <= ScreenTrans - 1; i = i + ifactor)
3658     for (j = 0; j <= ScreenLong - 1; j = j + ifactor)
3659     {
3660         if (ScreenFlag[i][j] == true)
3661         {
3662             R88[i][j] = ScreenR_Most[i][j];
3663             G88[i][j] = ScreenG_Most[i][j];
3664             B88[i][j] = ScreenB_Most[i][j];
3665         }
3666     }

```

```

3651     }
3652 }
3653
3654 for (i = 0; i <= ScreenTrans - 1; i++)
3655     for (j = 0; j <= ScreenLong - 1; j++)
3656         ScreenFullFlag[i][j] = false;
3657
3658 for (i = 0; i <= ScreenTrans - 1; i = i + ifactor)
3659     for (j = 0; j <= ScreenLong - 1; j = j + ifactor)
3660     {
3661         if (ScreenFlag[i][j] == true)
3662         {
3663             ScreenFullFlag[i * ifactor][j * ifactor] = true;
3664             R88[i * ifactor][j * ifactor] = ScreenR_Most[i][j];
3665             G88[i * ifactor][j * ifactor] = ScreenG_Most[i][j];
3666             B88[i * ifactor][j * ifactor] = ScreenB_Most[i][j];
3667         }
3668     }
3669
3670 for (i = 0; i <= ScreenTrans - 1; i++)
3671     for (j = 0; j <= ScreenLong - 1; j++)
3672     {
3673         LongNum = static_cast<long>(i) * static_cast<long>(ScreenLong) +
3674             static_cast<long>(j);
3675         ScreenFullFlag_1[LongNum] = 0;
3676         R88_1[LongNum] = 0;
3677         G88_1[LongNum] = 0;
3678         B88_1[LongNum] = 0;
3679     }
3680
3681 for (i = 0; i <= ScreenTrans - 1; i++)
3682     for (j = 0; j <= ScreenLong - 1; j++)
3683     {
3684         LongNum = static_cast<long>(i) * static_cast<long>(ScreenLong) +
3685             static_cast<long>(j);
3686         if (ScreenFullFlag[i][j] == true)
3687         {
3688             ScreenFullFlag_1[LongNum] = 1;
3689             R88_1[LongNum] = R88[i][j];
3690             G88_1[LongNum] = G88[i][j];
3691             B88_1[LongNum] = B88[i][j];
3692         }
3693         else
3694         {
3695             R88_1[LongNum] = 0;
3696             G88_1[LongNum] = 0;
3697             B88_1[LongNum] = 0;
3698         }
3699     }
3700
3701 return 1;
3702 }
3703
3704 void XY2XXY(double epsiron, double x0, double& x11, double y0, double& y1)
3705 {
3706     x11 = cos(epsiron) * x0 - sin(epsiron) * y0;
3707     y1 = sin(epsiron) * x0 + cos(epsiron) * y0;
3708 }
3709
3710 int ConvertPhoto2Circle2(int ScreenTrans, int ScreenLong, int ScreenOrigTrans, int
    ScreenOrigLong, bool FlagInv, double& XXX, double& ZZZ, double** LinHomG, double**
    LinIHomG)

```

```

3711 {
3712     double Uout, Vout;
3713     double h9;
3714
3715     if (abs(XXX) < 0.0000000001) XXX = 0.0;
3716
3717     if (abs(ZZZ) < 0.0000000001) ZZZ = 0.0;
3718
3719     if (FlagInv == false)
3720     {
3721         Uout = LinHomG[1][1] * XXX + LinHomG[1][2] * ZZZ + LinHomG[1][3] * 1.0;
3722         Vout = LinHomG[2][1] * XXX + LinHomG[2][2] * ZZZ + LinHomG[2][3] * 1.0;
3723         h9 = LinHomG[3][1] * XXX + LinHomG[3][2] * ZZZ + LinHomG[3][3] * 1.0;
3724
3725         XXX = Uout / h9 * ScreenTrans / ScreenOrigTrans;
3726         ZZZ = Vout / h9 * ScreenLong / ScreenOrigLong;
3727     }
3728
3729     else
3730     {
3731         XXX = XXX / ScreenTrans * ScreenOrigTrans;
3732         ZZZ = ZZZ / ScreenLong * ScreenOrigLong;
3733         Uout = LinHomG[1][1] * XXX + LinHomG[1][2] * ZZZ + LinHomG[1][3] * 1.0;
3734         Vout = LinHomG[2][1] * XXX + LinHomG[2][2] * ZZZ + LinHomG[2][3] * 1.0;
3735         h9 = LinHomG[3][1] * XXX + LinHomG[3][2] * ZZZ + LinHomG[3][3] * 1.0;
3736
3737         XXX = Uout / h9;
3738         ZZZ = Vout / h9;
3739     }
3740
3741     return 1;
3742 }
3743
3744
3745
3746 int IrrRefBottom(double* R_Bot_2d_1, double* G_Bot_2d_1, double* B_Bot_2d_1, double*
Params, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int
MaxY, double* bw, double* Absorbance, double* f_fine, double MieC)
3747 {
3748     int i, j;
3749     double x11, y11, z11;
3750     double AA;
3751     double Rtemp, Gtemp, Btemp;
3752     bool Hitpath0;
3753     double b11x, b11y, b11z;
3754     double x33, y33;
3755     double s2;
3756     double x4888, y4888;
3757     double L1, L2;
3758     double thetadash7;
3759     double reffactor22;
3760     double Pi;
3761     double alfa, beta;
3762     double Xsum, Ysum, Zsum;
3763     int n;
3764     double Exp1;
3765     double R_val, G_val, B_val;
3766     double Axdash, Aydash, Azdash;
3767     double X_gakeue, Y_gakeue;
3768     long i1, j1;
3769     double xtemp, ytemp;
3770     double Gauss;
3771     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,
FacDetailed, LakeShoreEPS;

```



```

3772     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,
        Rittaikaku, refl;
3773
3774     double Eobs;
3775
3776     double y0;
3777
3778     double iLambertian;
3779
3780     double dS;
3781
3782     FactorDC = Params[0];
3783     gzai = Params[1];
3784     sigma = Params[2];
3785     delta = Params[3];
3786     deltaLamda = Params[4];
3787     r1 = Params[5];
3788     d = Params[6];
3789     Xgrid = Params[7];
3790     Ygrid = Params[8];
3791     FacDetailed = Params[9];
3792     LakeShoreEPS = Params[10];
3793     fai77 = Params[11];
3794     theta77 = Params[12];
3795     n12 = Params[13];
3796     xobs = Params[14];
3797     yobs = Params[15];
3798     zobs = Params[16];
3799     k1 = Params[17];
3800     Km = Params[18];
3801     SettleShodo = Params[19];
3802     TheoShodo = Params[20];
3803     Rittaikaku = Params[21];
3804     refl = Params[22];
3805     y0 = Params[23];
3806     iLambertian = Params[24];
3807
3808     double** R_Bot_2d;
3809     double** G_Bot_2d;
3810     double** B_Bot_2d;
3811
3812     R_Bot_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
3813     G_Bot_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
3814     B_Bot_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
3815
3816     for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1; i1++)
3817         for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
            (FacDetailed) + 1; j1++)
3818         {
3819             R_Bot_2d[i1][j1] = 0.0;
3820             G_Bot_2d[i1][j1] = 0.0;
3821             B_Bot_2d[i1][j1] = 0.0;
3822         }
3823
3824     for (i1 = 0; i1 <= (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1) * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1); i1++)

```

```

3825     {
3826         R_Bot_2d_1[i1] = 0.0;
3827         G_Bot_2d_1[i1] = 0.0;
3828         B_Bot_2d_1[i1] = 0.0;
3829     }
3830
3831     Pi = 3.14159265358979;
3832
3833     AA = r1 / sqrt(d);
3834
3835     double theta77_after;
3836     double theta_at_bottom;
3837     double w1;
3838     double dummy1;
3839     double Housenvector1[5];
3840     double Resultvector1[5];
3841     double Nyushavector1[5];
3842
3843     for (i=0; i < 5; i++)
3844     {
3845         Housenvector1[i] = 0.0;
3846         Resultvector1[i] = 0.0;
3847         Nyushavector1[i] = 0.0;
3848     }
3849
3850     for (i = 0; i <= MaxX; i++)
3851     for (j = 0; j <= MaxY; j++)
3852     {
3853
3854         x11 = -r1 + static_cast<double>(i) * Xgrid;
3855         y11 = -r1 + static_cast<double>(j) * Ygrid;
3856         z11 = pow(x11, 2) / pow(AA, 2) + pow(y11, 2) / pow(AA, 2) - d;
3857
3858         if ((pow(x11, 2) + pow(y11, 2) - pow(r1, 2) > -LakeShoreEPS) || (z11 >= -
            LakeShoreEPS)) goto nocalc000;
3859
3860         Rtemp = 0.0;
3861         Gtemp = 0.0;
3862         Btemp = 0.0;
3863
3864         Hitpath0 = true;
3865
3866         Housenvector1[1] = -2.0 * x11;
3867         Housenvector1[2] = -2.0 * y11;
3868         Housenvector1[3] = AA * AA;
3869
3870         dummy1 = sqrt(pow(Housenvector1[1], 2) + pow(Housenvector1[2], 2) + pow
            (Housenvector1[3], 2));
3871         Housenvector1[1] = Housenvector1[1] / dummy1;
3872         Housenvector1[2] = Housenvector1[2] / dummy1;
3873         Housenvector1[3] = Housenvector1[3] / dummy1;
3874
3875         Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
3876         Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
3877         if (1.0 - 1.0 / (pow(n12, 2)) * pow(cos(theta77), 2) < 0.0)
3878         {
3879             Hitpath0 = false;
3880         }
3881         else
3882         {
3883             Resultvector1[3] = -sqrt(1.0 - 1.0 / (pow(n12, 2)) * pow(cos
                (theta77), 2));
3884         }
3885     }

```

```

3886     if (Hitpath0 == false) goto nocalc000;
3887     theta77_after = acos(sqrt(1.0 - 1.0 / (pow(n12, 2)) * pow(cos(theta77),
3888                                     2))); //20250505
3889
3889     Nyushavector1[1] = -Resultvector1[1];
3890     Nyushavector1[2] = -Resultvector1[2];
3891     Nyushavector1[3] = -Resultvector1[3];
3892
3893     dummy1 = sqrt(pow(Nyushavector1[1], 2) + pow(Nyushavector1[2], 2) + pow
3894               (Nyushavector1[3], 2));
3894     Nyushavector1[1] = Nyushavector1[1] / dummy1;
3895     Nyushavector1[2] = Nyushavector1[2] / dummy1;
3896     Nyushavector1[3] = Nyushavector1[3] / dummy1;
3897
3898     dummy1 = Nyushavector1[1] * Housenvector1[1] + Nyushavector1[2] *
3899               Housenvector1[2] + Nyushavector1[3] * Housenvector1[3];
3900     theta_at_bottom = acos(dummy1); //20250505
3901
3902
3903     b11x = 0.0;
3904     b11y = 0.0;
3905     b11z = 0.0;
3906
3907     Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, z11, xobs, yobs,
3908                   zobs, n12);
3909
3909     if (Hitpath0 == false) goto nocalc000;
3910
3911     x33 = x11 + z11 / sqrt(1.0 - (pow(cos(theta77) / n12, 2))) * cos(
3912               theta77) * cos(fai77) / n12;
3912     y33 = y11 - z11 / sqrt(1.0 - (pow(cos(theta77) / n12, 2))) * cos(
3913               theta77) * sin(fai77) / n12;
3914
3914     if (b11z == 0.0)
3915     {
3916         Hitpath0 = false;
3917         goto nocalc000;
3918     }
3919
3920     s2 = (0.0 - z11) / b11z;
3921     x4888 = x11 + s2 * b11x;
3922     y4888 = y11 + s2 * b11y;
3923
3924
3925     L1 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(z11, 2));
3926     L2 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(z11, 2));
3927
3928     if (Hitpath0 == false) goto nocalc000;
3929
3930     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
3931     reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin(
3932               theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan(
3933               theta77 + thetadash7), 2)));
3932     beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(y4888 -
3933               yobs, 2)));
3934     alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
3935               y4888, 2)));
3934     reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin(
3936               beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
3937
3937     if (Hitpath0 == false) goto nocalc000;
3938
3938     dS = Xgrid * Ygrid;

```

```

3939
3940     Xsum = 0.0;
3941     Ysum = 0.0;
3942     Zsum = 0.0;
3943
3944
3945     for (n = 1; n <= 70; n++)
3946     {
3947         wl = 380.0 + static_cast<double>(n - 1) * 5.0;
3948         Exp1 = exp(-(L1 + L2) * (k1 + bw[n] + Absorbance[n] + MieC*650.0/
3949             wl));
3950
3951         //Eobs = Km * f_fine[n] * SettleShodo / TheoShodo * refractor22 *
3952             Exp1 * refl / Pi * dS / (n12 * n12)* b11z;
3953         Eobs = 1.0 / cos(theta77_after) * cos(theta_at_bottom) * Km * f_fine
3954             [n] * SettleShodo / TheoShodo * refractor22 * Exp1 * refl / Pi *
3955             dS / (n12 * n12) * b11z;
3956
3957         Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
3958         Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
3959         Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
3960     }
3961
3962     R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
3963     G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
3964     B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
3965
3966     if (R_val < 0.0) R_val = 0.0;
3967     else R_val = R_val * FactorDC;
3968
3969     if (G_val < 0.0) G_val = 0.0;
3970     else G_val = G_val * FactorDC;
3971
3972     if (B_val < 0.0) B_val = 0.0;
3973     else B_val = B_val * FactorDC;
3974
3975     if (pow(x33, 2) + pow(y33, 2) >= pow(r1, 2)) Hitpath0 = false;
3976     if (Hitpath0 == false) goto nocalc000;
3977
3978     if (FlagCliff == 1)
3979     {
3980         Axdash = cos(theta77) * cos(fai77);
3981         Aydash = -cos(theta77) * sin(fai77);
3982         Azdash = -sin(theta77);
3983
3984         X_gakeue = x33 + gzai / Azdash * Axdash;
3985         Y_gakeue = y33 + gzai / Azdash * Aydash;
3986
3987         if (sqrt(pow(X_gakeue, 2) + pow(Y_gakeue, 2)) > r1 + gzai / (tan
3988             (delta))) Hitpath0 = false;
3989     }
3990
3991     if (pow(x4888, 2) + pow(y4888, 2) > pow(r1, 2)) Hitpath0 = false;
3992
3993     if (Hitpath0 == false) goto nocalc000;
3994
3995     Rtemp = R_val;
3996     if (Rtemp < 0.0) Rtemp = 0.0;
3997     Gtemp = G_val;
3998     if (Gtemp < 0.0) Gtemp = 0.0;
3999     Btemp = B_val;
4000     if (Btemp < 0.0) Btemp = 0.0;

```

```

3998     for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1; i1++)
3999         for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) *
static_cast<long>(FacDetailed) + 1; j1++)
4000         {
4001             xtemp = static_cast<double>(i1) / FacDetailed - r1;
4002             ytemp = static_cast<double>(j1) / FacDetailed - r1;
4003             if (pow(xtemp, 2) + pow(ytemp, 2) > pow(r1, 2))
4004             {
4005                 goto NotSuper000;
4006             }
4007             else
4008             {
4009                 Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow(x4888 -
xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 * pow(sigma,
2)));
4010                 R_Bot_2d[i1][j1] = R_Bot_2d[i1][j1] + Rtemp * Gauss;
4011                 G_Bot_2d[i1][j1] = G_Bot_2d[i1][j1] + Gtemp * Gauss;
4012                 B_Bot_2d[i1][j1] = B_Bot_2d[i1][j1] + Btemp * Gauss;
4013             }
4014         }
4015         NotSuper000:
4016         ;
4017     }
4018     nocalc000:
4019     ;
4020 }
4021
4022 for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1; i1++)
4023     for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1; j1++)
4024     {
4025         R_Bot_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1) + j1] = R_Bot_2d[i1][j1];
4026         G_Bot_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1) + j1] = G_Bot_2d[i1][j1];
4027         B_Bot_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1) + j1] = B_Bot_2d[i1][j1];
4028     }
4029
4030     return 1;
4031 }
4032 }
4033
4034 int DenstFlucSca(double* R_ES_2d_1, double* G_ES_2d_1, double* B_ES_2d_1, double*
Params1, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int
MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double MieC)
4035 {
4036     int i, j;
4037     double x11, y11, z11;
4038     double AA;
4039     double Rtemp, Gtemp, Btemp;
4040     bool Hitpath0;
4041     double scale1;
4042     double Resultvector1[5];
4043     double b11x, b11y, b11z;
4044     double x33, y33;
4045     double s2;
4046     double x4888, y4888;
4047     double thetadash7;
4048     double refractor22;
4049     double Pi;
4050     double alfa, beta;

```

```

4051     double Xsum, Ysum, Zsum;
4052     int n;
4053     double Exp1;
4054     double R_val, G_val, B_val;
4055     double Axdash, Aydash, Azdash;
4056     double X_gakeue, Y_gakeue;
4057     long i1, j1;
4058     double xtemp, ytemp;
4059     double Gauss;
4060     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,      ↗
        FacDetailed, LakeShoreEPS;
4061     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo, ↗
        Rittaikaku, refl;
4062     int k;
4063     double zz;
4064     double FactorZ;
4065     double CosScatAng, beta90factor, delta0, PhaseFunc, L3, L4;
4066     double Eobs;
4067     double y0;
4068     double dS;
4069
4070     FactorDC = Params1[0];
4071     gzai = Params1[1];
4072     sigma = Params1[2];
4073     delta = Params1[3];
4074     deltaLamda = Params1[4];
4075     r1 = Params1[5];
4076     d = Params1[6];
4077     Xgrid = Params1[7];
4078     Ygrid = Params1[8];
4079     FacDetailed = Params1[9];
4080     LakeShoreEPS = Params1[10];
4081     fai77 = Params1[11];
4082     theta77 = Params1[12];
4083     n12 = Params1[13];
4084     xobs = Params1[14];
4085     yobs = Params1[15];
4086     zobs = Params1[16];
4087     k1 = Params1[17];
4088     Km = Params1[18];
4089     SettleShodo = Params1[19];
4090     TheoShodo = Params1[20];
4091     Rittaikaku = Params1[21];
4092     refl = Params1[22];
4093     delta0 = Params1[23];
4094     FactorZ = Params1[24];
4095     y0 = Params1[25];
4096
4097     double** R_ES_2d;
4098     double** G_ES_2d;
4099     double** B_ES_2d;
4100
4101
4102     Pi = 3.14159265358979;
4103
4104     R_ES_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1);
4105     G_ES_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1);
4106     B_ES_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>      ↗
        (FacDetailed) + 1);

```

```

4107
4108     for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1; i1++)
4109         for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1; j1++)
4110         {
4111             R_ES_2d[i1][j1] = 0.0;
4112             G_ES_2d[i1][j1] = 0.0;
4113             B_ES_2d[i1][j1] = 0.0;
4114         }
4115
4116     for (i1 = 0; i1 < (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1) * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
(FacDetailed) + 1); i1++)
4117     {
4118         R_ES_2d_1[i1] = 0.0;
4119         G_ES_2d_1[i1] = 0.0;
4120         B_ES_2d_1[i1] = 0.0;
4121     }
4122
4123     AA = r1 / sqrt(d);
4124
4125     double wl;
4126
4127     for (i = 0; i <= MaxX; i++)
4128         for (j = 0; j <= MaxY; j++)
4129             for (k = 0; k <= Maxk; k++)
4130             {
4131                 x11 = -r1 + static_cast<double>(i) * Xgrid;
4132                 y11 = -r1 + static_cast<double>(j) * Ygrid;
4133                 z11 = (x11 * x11) / (AA * AA) + (y11 * y11) / (AA * AA) - d;
4134
4135                 zz = -static_cast<double>(k) * FactorZ;
4136
4137                 if (pow(x11, 2) + pow(y11, 2) - pow(r1, 2) > -LakeShoreEPS) goto
nocalc200;
4138
4139                 if ((z11 - zz > -LakeShoreEPS) || (zz >= -LakeShoreEPS)) goto
nocalc200;
4140
4141                 Rtemp = 0.0;
4142                 Gtemp = 0.0;
4143                 Btemp = 0.0;
4144
4145                 Hitpath0 = true;
4146                 scale1 = 1.0;
4147
4148                 Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
4149                 Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
4150                 if (1.0 - 1.0 / (n12 * n12) * pow(cos(theta77), 2) < 0.0) Hitpath0 =
false;
4151                 else Resultvector1[3] = -sqrt(1.0 - 1.0 / (n12 * n12) * pow(cos
(theta77), 2));
4152
4153                 if (Hitpath0 == false) goto nocalc200;
4154
4155                 x33 = x11 + zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * cos(fai77) / n12;
4156                 y33 = y11 - zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
(theta77) * sin(fai77) / n12;
4157
4158                 if (Hitpath0 == false) goto nocalc200;
4159
4160                 b11x = 0.0;

```

```

4161         b11y = 0.0;
4162         b11z = 0.0;
4163
4164         Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, zz, xobs, yobs, zobs, n12);
4165
4166         if (Hitpath0 == false) goto nocalc200;
4167
4168         if (b11z == 0.0)
4169         {
4170             Hitpath0 = false;
4171             goto nocalc200;
4172         }
4173
4174         s2 = (0.0 - zz) / b11z;
4175         x4888 = x11 + s2 * b11x;
4176         y4888 = y11 + s2 * b11y;
4177
4178         thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
4179         refractor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin
4180             (theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan
4181             (theta77 + thetadash7), 2)));
4182         beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(
4183             y4888 - yobs, 2)));
4184         alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
4185             y4888, 2)));
4186         refractor22 = refractor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin
4187             (beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
4188
4189         CosScatAng = Resultvector1[1] * b11x + Resultvector1[2] * b11y +
4190             Resultvector1[3] * b11z;
4191         beta90factor = 1.0 / (8.0 * Pi / 3.0 * (2.0 + delta0) / (1.0 +
4192             delta0));
4193         PhaseFunc = beta90factor * (1.0 + (1.0 - delta0) / (1.0 + delta0) *
4194             pow(CosScatAng, 2));
4195         L3 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(zz, 2));
4196         L4 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(zz,
4197             2));
4198
4199         dS = Xgrid * Ygrid;
4200
4201         Xsum = 0.0;
4202         Ysum = 0.0;
4203         Zsum = 0.0;
4204
4205         for (n = 1; n <= 70; n++)
4206         {
4207             wl = 380.0 + static_cast<double>(n - 1) * 5.0;
4208             Exp1 = exp(-(L3 + L4) * (k1 + bw[n] + Absorbance[n] + MieC *
4209                 650.0 / wl));
4210             Eobs = Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
4211                 refractor22 * Exp1 * PhaseFunc * bw[n] * dS / (n12 * n12);
4212
4213             Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
4214             Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
4215             Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;
4216         }
4217
4218         R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
4219         G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
4220         B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
4221
4222         if (R_val < 0.0) R_val = 0.0;
4223         else R_val = R_val * FactorDC;

```



```

4213
4214         if (G_val < 0.0) G_val = 0.0;
4215         else G_val = G_val * FactorDC;
4216
4217         if (B_val < 0.0) B_val = 0.0;
4218         else B_val = B_val * FactorDC;
4219
4220         Rtemp = R_val;
4221         if (Rtemp < 0.0) Rtemp = 0.0;
4222         Gtemp = G_val;
4223         if (Gtemp < 0.0) Gtemp = 0.0;
4224         Btemp = B_val;
4225         if (Btemp < 0.0) Btemp = 0.0;
4226
4227         if (pow(x33, 2) + pow(y33, 2) >= pow(r1, 2)) Hitpath0 = false;
4228         if (Hitpath0 == false) goto nocalc200;
4229
4230         if (FlagCliff == 1)
4231         {
4232             Axdash = cos(theta77) * cos(fai77);
4233             Aydash = -cos(theta77) * sin(fai77);
4234             Azdash = -sin(theta77);
4235
4236             X_gakeue = x33 + gzai / Azdash * Axdash;
4237             Y_gakeue = y33 + gzai / Azdash * Aydash;
4238
4239             if (sqrt(pow(X_gakeue, 2) + pow(Y_gakeue, 2)) > r1 + gzai / (tan
4240                 (delta))) Hitpath0 = false;
4241         }
4242
4243         if (pow(x4888, 2) + pow(y4888, 2) > pow(r1, 2)) Hitpath0 = false;
4244         if (Hitpath0 == false) goto nocalc200;
4245
4246         for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) *
4247             static_cast<long>(FacDetailed) + 1; i1++)
4248             for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) *
4249                 static_cast<long>(FacDetailed) + 1; j1++)
4250             {
4251                 xtemp = static_cast<double>(i1) / FacDetailed - r1;
4252                 ytemp = static_cast<double>(j1) / FacDetailed - r1;
4253
4254                 if (pow(xtemp, 2) + pow(ytemp, 2) > pow(r1, 2))
4255                 {
4256                     goto NotSuper200;
4257                 }
4258                 else
4259                 {
4260                     Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow
4261                         (x4888 - xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 *
4262                         pow(sigma, 2)));
4263                     R_ES_2d[i1][j1] = R_ES_2d[i1][j1] + Rtemp * Gauss;
4264                     G_ES_2d[i1][j1] = G_ES_2d[i1][j1] + Gtemp * Gauss;
4265                     B_ES_2d[i1][j1] = B_ES_2d[i1][j1] + Btemp * Gauss;
4266                 }
4267             }
4268         NotSuper200:
4269         ;
4270     }
4271     nocalc200:
4272     ;
4273 }
4274
4275 for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>

```

```

    (FacDetailed) + 1; i1++)
4272     for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>  P
        (FacDetailed) + 1; j1++)
4273     {
4274         R_ES_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>  P
            (FacDetailed) + 1) + j1] = R_ES_2d[i1][j1];
4275         G_ES_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>  P
            (FacDetailed) + 1) + j1] = G_ES_2d[i1][j1];
4276         B_ES_2d_1[i1 * (static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>  P
            (FacDetailed) + 1) + j1] = B_ES_2d[i1][j1];
4277     }
4278
4279     return 1;
4280
4281 }
4282
4283 int MieSca(double* R_Mie_2d_1, double* G_Mie_2d_1, double* B_Mie_2d_1, double*  P
    Params1, int FlagCliff, double* Xbar, double* Ybar, double* Zbar, int MaxX, int  P
    MaxY, int Maxk, double* bw, double* Absorbance, double* f_fine, double Mieci)
4284 {
4285     int i, j;
4286     double x11, y11, z11;
4287     double AA;
4288     double Rtemp, Gtemp, Btemp;
4289     bool Hitpath0;
4290     double scale1;
4291     double Resultvector1[5];
4292     double b11x, b11y, b11z;
4293     double x33, y33;
4294     double s2;
4295     double x4888, y4888;
4296     double thetadash7;
4297     double reffactor22;
4298     double Pi;
4299     double alfa, beta;
4300     double Xsum, Ysum, Zsum;
4301     int n;
4302     double Exp1;
4303     double R_val, G_val, B_val;
4304     double Axdash, Aydash, Azdash;
4305     double X_gakeue, Y_gakeue;
4306     long i1, j1;
4307     double xtemp, ytemp;
4308     double Gauss;
4309     double FactorDC, gzai, sigma, delta, deltaLamda, r1, d, Xgrid, Ygrid,  P
        FacDetailed, LakeShoreEPS;
4310     double fai77, theta77, n12, xobs, yobs, zobs, k1, Km, SettleShodo, TheoShodo,  P
        Rittaikaku, refl;
4311     int k;
4312     double zz;
4313     double FactorZ;
4314     double CosScatAng, delta0, PhaseFunc, L3, L4;
4315     double Miea0, Miea1, Miea2, Miea3, Miea4;
4316     double ScatAng;
4317
4318     double Eobs;
4319
4320     double dummy1;
4321
4322     double iPetzoldPF;
4323     double a6, a5, a4, a3, a2, a1, a0;
4324
4325     double y0;
4326

```

```

4327     double dS;
4328
4329     FactorDC = Params1[0];
4330     gzai = Params1[1];
4331     sigma = Params1[2];
4332     delta = Params1[3];
4333     deltaLamda = Params1[4];
4334     r1 = Params1[5];
4335     d = Params1[6];
4336     Xgrid = Params1[7];
4337     Ygrid = Params1[8];
4338     FacDetailed = Params1[9];
4339     LakeShoreEPS = Params1[10];
4340     fai77 = Params1[11];
4341     theta77 = Params1[12];
4342     n12 = Params1[13];
4343     xobs = Params1[14];
4344     yobs = Params1[15];
4345     zobs = Params1[16];
4346     k1 = Params1[17];
4347     Km = Params1[18];
4348     SettleShodo = Params1[19];
4349     TheoShodo = Params1[20];
4350     Rittaikaku = Params1[21];
4351     refl = Params1[22];
4352     delta0 = Params1[23];
4353     FactorZ = Params1[24];
4354     Miew0 = Params1[25];
4355     Miew1 = Params1[26];
4356     Miew2 = Params1[27];
4357     Miew3 = Params1[28];
4358     Miew4 = Params1[29];
4359     iPetzoldPF = Params1[30];
4360     y0 = Params1[31];
4361
4362     Pi = 3.14159265358979;
4363
4364     a6 = 0.1571;
4365     a5 = -0.4145;
4366     a4 = -0.0804;
4367     a3 = 0.6345;
4368     a2 = -0.2644;
4369     a1 = -1.7688;
4370     a0 = 1.8269;
4371
4372     double** R_Mie_2d;
4373     double** G_Mie_2d;
4374     double** B_Mie_2d;
4375
4376     R_Mie_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
4377     G_Mie_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
4378     B_Mie_2d = alloc_matrix(static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1, static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1);
4379
4380     for (i1 = 0; i1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
        (FacDetailed) + 1; i1++)
4381         for (j1 = 0; j1 < static_cast<long>(2.0 * r1 + 1.0) * static_cast<long>
            (FacDetailed) + 1; j1++)
4382             {

```

```

4383         R_Mie_2d[i1][j1] = 0.0;
4384         G_Mie_2d[i1][j1] = 0.0;
4385         B_Mie_2d[i1][j1] = 0.0;
4386     }
4387
4388     for (i1 = 0; i1 <= (int)(2 * r1 + 1) * int(FacDetailed) + 1) * (int)(2 * r1 + 1) *
         int(FacDetailed) + 1); i1++)
4389     {
4390         R_Mie_2d_1[i1] = 0.0;
4391         G_Mie_2d_1[i1] = 0.0;
4392         B_Mie_2d_1[i1] = 0.0;
4393     }
4394
4395     Pi = 3.14159265358979;
4396
4397     AA = r1 / sqrt(d);
4398
4399     double wl;
4400
4401     for (i = 0; i <= MaxX; i++)
4402     for (j = 0; j <= MaxY; j++)
4403     for (k = 0; k <= Maxk; k++)
4404     {
4405         x11 = -r1 + static_cast<double>(i) * Xgrid;
4406         y11 = -r1 + static_cast<double>(j) * Ygrid;
4407         z11 = (x11 * x11) / (AA * AA) + (y11 * y11) / (AA * AA) - d;
4408
4409         zz = -static_cast<double>(k) * FactorZ;
4410
4411         if (pow(x11, 2) + pow(y11, 2) - pow(r1, 2) > -LakeShoreEPS) goto
            nocalc300;
4412
4413         if ((z11 - zz > -LakeShoreEPS) || (zz >= -LakeShoreEPS)) goto
            nocalc300;
4414
4415         Hitpath0 = true;
4416         scale1 = 1.0;
4417
4418         Resultvector1[1] = cos(theta77) * cos(fai77) / n12;
4419         Resultvector1[2] = -cos(theta77) * sin(fai77) / n12;
4420         if (1.0 - 1.0 / (n12 * n12) * pow(cos(theta77), 2) < 0.0) Hitpath0 =
            false;
4421         else Resultvector1[3] = -sqrt(1.0 - 1.0 / (n12 * n12) * pow(cos
            (theta77), 2));
4422
4423         if (Hitpath0 == false) goto nocalc300;
4424
4425         x33 = x11 + zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
            (theta77) * cos(fai77) / n12;
4426         y33 = y11 - zz / sqrt(1 - (pow(cos(theta77) / n12, 2))) * cos
            (theta77) * sin(fai77) / n12;
4427
4428         if (Hitpath0 == false) goto nocalc300;
4429
4430         Rtemp = 0.0;
4431         Gtemp = 0.0;
4432         Btemp = 0.0;
4433
4434         b11x = 0.0;
4435         b11y = 0.0;
4436         b11z = 0.0;
4437
4438         Solve4thEq_DLL(Hitpath0, b11x, b11y, b11z, x11, y11, zz, xobs, yobs,

```

```

zobs, n12);
4440
4441     if (Hitpath0 == false) goto nocalc300;
4442
4443     if (b11z == 0)
4444     {
4445         Hitpath0 = false;
4446         goto nocalc300;
4447     }
4448
4449     s2 = (0.0 - zz) / b11z;
4450     x4888 = x11 + s2 * b11x;
4451     y4888 = y11 + s2 * b11y;
4452
4453     thetadash7 = asin(sqrt(1.0 - (pow(cos(theta77), 2) / pow(n12, 2))));
4454     reffactor22 = (1.0 - 0.5 * (pow(sin(theta77 - thetadash7) / sin
        (theta77 + thetadash7), 2) + pow(tan(theta77 - thetadash7) / tan
        (theta77 + thetadash7), 2)));
4455     beta = Pi / 2.0 - atan2(zobs, sqrt(pow(x4888 - xobs, 2) + pow(
        y4888 - yobs, 2)));
4456     alfa = Pi / 2.0 - atan2(-z11, sqrt(pow(x11 - x4888, 2) + pow(y11 -
        y4888, 2)));
4457     reffactor22 = reffactor22 * (1.0 - 0.5 * (pow(sin(beta - alfa) / sin
        (beta + alfa), 2) + pow(tan(beta - alfa) / tan(beta + alfa), 2)));
4458
4459     CosScatAng = Resultvector1[1] * b11x + Resultvector1[2] * b11y +
        Resultvector1[3] * b11z;
4460     ScatAng = acos(CosScatAng) / Pi * 180;
4461
4462     if (fabs(iPetzoldPF - 1) < 0.01)
4463     {
4464         dummy1 = a6 * pow(log10(ScatAng), 6) + a5 * pow(log10(ScatAng),
            5) + a4 * pow(log10(ScatAng), 4) + a3 * pow(log10(ScatAng), 3)
            + a2 * pow(log10(ScatAng), 2) + a1 * log10(ScatAng) + a0;
4465         PhaseFunc = pow(10.0, dummy1);
4466     }
4467     else
4468     {
4469         PhaseFunc = Miea0 * pow(ScatAng, 4) + Miea1 * pow(ScatAng, 3) +
            Miea2 * pow(ScatAng, 2) + Miea3 * pow(ScatAng, 1) + Miea4;
4470         PhaseFunc = 0.0;
4471     }
4472
4473     L3 = sqrt(pow((x33 - x11), 2) + pow((y33 - y11), 2) + pow(zz, 2));
4474     L4 = sqrt(pow((x4888 - x11), 2) + pow((y4888 - y11), 2) + pow(zz,
        2));
4475
4476     dS = Xgrid * Ygrid;
4477
4478     Xsum = 0.0;
4479     Ysum = 0.0;
4480     Zsum = 0.0;
4481
4482     for (n = 1; n <= 70; n++)
4483     {
4484         wl = 380.0 + static_cast<double>(n - 1) * 5.0;
4485         Exp1 = exp(-(L3 + L4) * (k1 + bw[n] + Absorbance[n] + Miec *
            650.0 / wl));
4486         Eobs = Km * f_fine[n] * SettleShodo / TheoShodo * FactorZ *
            reffactor22 * Exp1 * PhaseFunc * Miec * 650.0/wl*dS / (n12 *
            n12);
4487         Xsum = Xsum + Eobs * Xbar[n] * deltaLamda;
4488         Ysum = Ysum + Eobs * Ybar[n] * deltaLamda;
4489         Zsum = Zsum + Eobs * Zbar[n] * deltaLamda;

```

```

4490     }
4491
4492     R_val = 3.241 * Xsum - 1.5374 * Ysum - 0.4986 * Zsum;
4493     G_val = -0.9692 * Xsum + 1.876 * Ysum + 0.0416 * Zsum;
4494     B_val = 0.0556 * Xsum - 0.204 * Ysum + 1.057 * Zsum;
4495
4496     if (R_val < 0.0) R_val = 0.0;
4497     else R_val = R_val * FactorDC;
4498
4499     if (G_val < 0.0) G_val = 0.0;
4500     else G_val = G_val * FactorDC;
4501
4502     if (B_val < 0.0) B_val = 0.0;
4503     else B_val = B_val * FactorDC;
4504
4505     Rtemp = R_val;
4506     if (Rtemp < 0.0) Rtemp = 0.0;
4507     Gtemp = G_val;
4508     if (Gtemp < 0.0) Gtemp = 0.0;
4509     Btemp = B_val;
4510     if (Btemp < 0.0) Btemp = 0.0;
4511
4512     if (pow(x33, 2) + pow(y33, 2) >= pow(r1, 2)) Hitpath0 = false;
4513     if (Hitpath0 == false) goto nocalc300;
4514
4515     if (FlagCliff == 1)
4516     {
4517         Axdash = cos(theta77) * cos(fai77);
4518         Aydash = -cos(theta77) * sin(fai77);
4519         Azdash = -sin(theta77);
4520
4521         X_gakeue = x33 + gzai / Azdash * Axdash;
4522         Y_gakeue = y33 + gzai / Azdash * Aydash;
4523
4524         if (sqrt(pow(X_gakeue, 2) + pow(Y_gakeue, 2)) > r1 + gzai / (tan
4525             (delta))) Hitpath0 = false;
4526     }
4527
4528     if (pow(x4888, 2) + pow(y4888, 2) > pow(r1, 2)) Hitpath0 = false;
4529
4530     if (Hitpath0 == false) goto nocalc300;
4531
4532     for (i1 = 0; i1 <= int(2 * r1 + 1) * FacDetailed; i1++)
4533     {
4534         for (j1 = 0; j1 <= int(2 * r1 + 1) * FacDetailed; j1++)
4535         {
4536             xtemp = static_cast<double>(i1) / FacDetailed - r1;
4537             ytemp = static_cast<double>(j1) / FacDetailed - r1;
4538
4539             if (pow(xtemp, 2) + pow(ytemp, 2) > pow(r1, 2))
4540             {
4541                 goto NotSuper300;
4542             }
4543             else
4544             {
4545                 Gauss = 1.0 / (2.0 * Pi * pow(sigma, 2)) * exp(-(pow
4546                     (x4888 - xtemp, 2) + pow(y4888 - ytemp, 2)) / (2.0 *
4547                     pow(sigma, 2)));
4548                 R_Mie_2d[i1][j1] = R_Mie_2d[i1][j1] + Rtemp * Gauss;
4549                 G_Mie_2d[i1][j1] = G_Mie_2d[i1][j1] + Gtemp * Gauss;
4550                 B_Mie_2d[i1][j1] = B_Mie_2d[i1][j1] + Btemp * Gauss;
4551             }
4552         }
4553     }
4554     NotSuper300:
4555     ;
4556 }

```

```

4551         nocalc300:
4552         ;
4553     }
4554
4555     for (i1 = 0; i1 <= int(2 * r1 + 1) * int(FacDetailed); i1++)
4556     for (j1 = 0; j1 <= int(2 * r1 + 1) * int(FacDetailed); j1++)
4557     {
4558         R_Mie_2d_1[i1 * (int(2 * r1 + 1) * int(FacDetailed) + 1) + j1] = R_Mie_2d_1
4559         [i1][j1];
4560         G_Mie_2d_1[i1 * (int(2 * r1 + 1) * int(FacDetailed) + 1) + j1] = G_Mie_2d_1
4561         [i1][j1];
4562         B_Mie_2d_1[i1 * (int(2 * r1 + 1) * int(FacDetailed) + 1) + j1] = B_Mie_2d_1
4563         [i1][j1];
4564     }
4565     return 1;
4566 }
4567
4568 double CalcFuresnel(double theta_In, double theta_Out)
4569 {
4570     double FuresnelInOut;
4571     FuresnelInOut = 1.0 - 0.5 * (pow(sin(theta_In - theta_Out) / sin(theta_In +
4572     theta_Out), 2) + pow(tan(theta_In - theta_Out) / tan(theta_In + theta_Out),
4573     2));
4574     return FuresnelInOut;
4575 }
4576 void Solve4thEq_DLL(bool& Hitpath111, double& b11x, double& b11y, double& b11z,
4577     double x11, double y11, double z11, double x41, double y41, double z41, double
4578     n12)
4579 {
4580     double Coeff[11];
4581     double SolutionReal[11], SolutionImag[11];
4582     double c0, c1, c2, c3, c4;
4583     int response;
4584     int i;
4585     double X;
4586
4587     Hitpath111 = true;
4588
4589     c4 = pow(z41, 4) - 2 * pow(z11, 2) * pow(z41, 2) + 2 * pow(y41, 2) * pow(z41, 2)
4590     - 4 * y11 * y41 * pow(z41, 2) + 2 * pow(y11, 2) * pow(z41, 2) + 2 * pow(x41,
4591     2) * pow(z41, 2) - 4 * x11 * x41 * pow(z41, 2) + 2 * pow(x11, 2) * pow(z41, 2)
4592     + pow(z11, 4) + 2 * pow(y41, 2) * pow(z11, 2);
4593     c4 = c4 - 4 * y11 * y41 * pow(z11, 2) + 2 * pow(y11, 2) * pow(z11, 2) + 2 * pow
4594     (x41, 2) * pow(z11, 2) - 4 * x11 * x41 * pow(z11, 2) + 2 * pow(x11, 2) * pow
4595     (z11, 2) + pow(y41, 4) - 4 * y11 * pow(y41, 3) + 6 * pow(y11, 2) * pow(y41, 2)
4596     + 2 * pow(x41, 2) * pow(y41, 2) - 4 * x11 * x41 * pow(y41, 2);
4597     c4 = c4 + 2 * pow(x11, 2) * pow(y41, 2) - 4 * pow(y11, 3) * y41 - 4 * pow(x41,
4598     2) * y11 * y41 + 8 * x11 * x41 * y11 * y41 - 4 * pow(x11, 2) * y11 * y41 + pow
4599     (y11, 4) + 2 * pow(x41, 2) * pow(y11, 2) - 4 * x11 * x41 * pow(y11, 2) + 2 *
4600     pow(x11, 2) * pow(y11, 2) + pow(x41, 4);
4601     c4 = c4 - 4 * x11 * pow(x41, 3) + 6 * pow(x11, 2) * pow(x41, 2) - 4 * pow(x11,
4602     3) * x41 + pow(x11, 4);
4603
4604     c3 = (-4 * pow(z41, 4) / pow(n12, 2)) + 2 * pow(z41, 4) + (6 * pow(z11, 2) *
4605     pow(z41, 2) / pow(n12, 2) - 2 * pow(z11, 2) * pow(z41, 2) - (6 * pow(y41,
4606     2) * pow(z41, 2) / pow(n12, 2) + 4 * pow(y41, 2) * pow(z41, 2);
4607     c3 = c3 + (12 * y11 * y41 * pow(z41, 2) / pow(n12, 2) - 8 * y11 * y41 * pow(
4608     z41, 2) - (6 * pow(y11, 2) * pow(z41, 2) / pow(n12, 2) + 4 * pow(y11, 2) * pow

```

```

(z41, 2) - (6 * pow(x41, 2) * pow(z41, 2)) / pow(n12, 2);
4595 c3 = c3 + 4 * pow(x41, 2) * pow(z41, 2) + (12 * x11 * x41 * pow(z41, 2)) / pow
      (n12, 2) - 8 * x11 * x41 * pow(z41, 2) - (6 * pow(x11, 2) * pow(z41, 2)) / pow
      (n12, 2) + 4 * pow(x11, 2) * pow(z41, 2) - (2 * pow(z11, 4)) / pow(n12, 2);
4596 c3 = c3 - (4 * pow(y41, 2) * pow(z11, 2)) / pow(n12, 2) + 2 * pow(y41, 2) * pow
      (z11, 2) + (8 * y11 * y41 * pow(z11, 2)) / pow(n12, 2) - 4 * y11 * y41 * pow
      (z11, 2) - (4 * pow(y11, 2) * pow(z11, 2)) / pow(n12, 2) + 2 * pow(y11, 2) *
      pow(z11, 2);
4597 c3 = c3 - (4 * pow(x41, 2) * pow(z11, 2)) / pow(n12, 2) + 2 * pow(x41, 2) * pow
      (z11, 2) + (8 * x11 * x41 * pow(z11, 2)) / pow(n12, 2) - 4 * x11 * x41 * pow
      (z11, 2) - (4 * pow(x11, 2) * pow(z11, 2)) / pow(n12, 2) + 2 * pow(x11, 2) *
      pow(z11, 2);
4598 c3 = c3 - (2 * pow(y41, 4)) / pow(n12, 2) + 2 * pow(y41, 4) + (8 * y11 * pow(
      y41, 3)) / pow(n12, 2) - 8 * y11 * pow(y41, 3) - (12 * pow(y11, 2) * pow(y41,
      2)) / pow(n12, 2) + 12 * pow(y11, 2) * pow(y41, 2);
4599 c3 = c3 - (4 * pow(x41, 2) * pow(y41, 2)) / pow(n12, 2) + 4 * pow(x41, 2) * pow
      (y41, 2) + (8 * x11 * x41 * pow(y41, 2)) / pow(n12, 2) - 8 * x11 * x41 * pow
      (y41, 2) - (4 * pow(x11, 2) * pow(y41, 2)) / pow(n12, 2);
4600 c3 = c3 + 4 * pow(x11, 2) * pow(y41, 2) + (8 * pow(y11, 3) * y41) / pow(n12,
      2) - 8 * pow(y11, 3) * y41 + (8 * pow(x41, 2) * y11 * y41) / pow(n12, 2) - 8 *
      pow(x41, 2) * y11 * y41 - (16 * x11 * x41 * y11 * y41) / pow(n12, 2);
4601 c3 = c3 + 16 * x11 * x41 * y11 * y41 + (8 * pow(x11, 2) * y11 * y41) / pow(n12,
      2) - 8 * pow(x11, 2) * y11 * y41 - (2 * pow(y11, 4)) / pow(n12, 2) + 2 * pow
      (y11, 4) - (4 * pow(x41, 2) * pow(y11, 2)) / pow(n12, 2) + 4 * pow(x41, 2) *
      pow(y11, 2);
4602 c3 = c3 + (8 * x11 * x41 * pow(y11, 2)) / pow(n12, 2) - 8 * x11 * x41 * pow(y11,
      2) - (4 * pow(x11, 2) * pow(y11, 2)) / pow(n12, 2);
4603 c3 = c3 + 4 * pow(x11, 2) * pow(y11, 2) - (2 * pow(x41, 4)) / pow(n12, 2) + 2 *
      pow(x41, 4) + (8 * x11 * pow(x41, 3)) / pow(n12, 2) - 8 * x11 * pow(x41, 3) -
      (12 * pow(x11, 2) * pow(x41, 2)) / pow(n12, 2) + 12 * pow(x11, 2) * pow(x41,
      2);
4604 c3 = c3 + (8 * pow(x11, 3) * x41) / pow(n12, 2) - 8 * pow(x11, 3) * x41 - (2 *
      pow(x11, 4)) / pow(n12, 2) + 2 * pow(x11, 4);
4605
4606 c2 = (- (6 * pow(z41, 4)) / pow(n12, 2)) + (6 * pow(z41, 4)) / pow(n12, 4) + pow
      (z41, 4) + (4 * pow(z11, 2) * pow(z41, 2)) / pow(n12, 2);
4607 c2 = c2 - (6 * pow(z11, 2) * pow(z41, 2)) / pow(n12, 4) - (8 * pow(y41, 2) * pow
      (z41, 2)) / pow(n12, 2) + (6 * pow(y41, 2) * pow(z41, 2)) / pow(n12, 4);
4608 c2 = c2 + 2 * pow(y41, 2) * pow(z41, 2) + (16 * y11 * y41 * pow(z41, 2)) / pow
      (n12, 2) - (12 * y11 * y41 * pow(z41, 2)) / pow(n12, 4);
4609 c2 = c2 - 4 * y11 * y41 * pow(z41, 2) - (8 * pow(y11, 2) * pow(z41, 2)) / pow
      (n12, 2) + (6 * pow(y11, 2) * pow(z41, 2)) / pow(n12, 4) + 2 * pow(y11, 2) *
      pow(z41, 2);
4610 c2 = c2 - (8 * pow(x41, 2) * pow(z41, 2)) / pow(n12, 2) + (6 * pow(x41, 2) * pow
      (z41, 2)) / pow(n12, 4) + 2 * pow(x41, 2) * pow(z41, 2) + (16 * x11 * x41 * pow
      (z41, 2)) / pow(n12, 2);
4611 c2 = c2 - (12 * x11 * x41 * pow(z41, 2)) / pow(n12, 4) - 4 * x11 * x41 * pow(
      z41, 2) - (8 * pow(x11, 2) * pow(z41, 2)) / pow(n12, 2);
4612 c2 = c2 + (6 * pow(x11, 2) * pow(z41, 2)) / pow(n12, 4) + 2 * pow(x11, 2) * pow
      (z41, 2) + pow(z11, 4) / pow(n12, 4) - (2 * pow(y41, 2) * pow(z11, 2)) / pow
      (n12, 2);
4613 c2 = c2 + (2 * pow(y41, 2) * pow(z11, 2)) / pow(n12, 4) + (4 * y11 * y41 * pow
      (z11, 2)) / pow(n12, 2) - (4 * y11 * y41 * pow(z11, 2)) / pow(n12, 4);
4614 c2 = c2 - (2 * pow(y11, 2) * pow(z11, 2)) / pow(n12, 2) + (2 * pow(y11, 2) * pow
      (z11, 2)) / pow(n12, 4) - (2 * pow(x41, 2) * pow(z11, 2)) / pow(n12, 2);
4615 c2 = c2 + (2 * pow(x41, 2) * pow(z11, 2)) / pow(n12, 4) + (4 * x11 * x41 * pow
      (z11, 2)) / pow(n12, 2) - (4 * x11 * x41 * pow(z11, 2)) / pow(n12, 4);
4616 c2 = c2 - (2 * pow(x11, 2) * pow(z11, 2)) / pow(n12, 2) + (2 * pow(x11, 2) * pow
      (z11, 2)) / pow(n12, 4) - (2 * pow(y41, 4)) / pow(n12, 2);
4617 c2 = c2 + pow(y41, 4) / pow(n12, 4) + pow(y41, 4) + (8 * y11 * pow(y41, 3)) / pow
      (n12, 2) - (4 * y11 * pow(y41, 3)) / pow(n12, 4);
4618 c2 = c2 - 4 * y11 * pow(y41, 3) - (12 * pow(y11, 2) * pow(y41, 2)) / pow(n12, 2)
      + (6 * pow(y11, 2) * pow(y41, 2)) / pow(n12, 4);
4619 c2 = c2 + 6 * pow(y11, 2) * pow(y41, 2) - (4 * pow(x41, 2) * pow(y41, 2)) / pow

```



```

        (n12, 2) + (2 * pow(x41, 2) * pow(y41, 2)) / pow(n12, 4);
4620    c2 = c2 + 2 * pow(x41, 2) * pow(y41, 2) + (8 * x11 * x41 * pow(y41, 2)) / pow
        (n12, 2) - (4 * x11 * x41 * pow(y41, 2)) / pow(n12, 4) - 4 * x11 * x41 * pow
        (y41, 2);
4621    c2 = c2 - (4 * pow(x11, 2) * pow(y41, 2)) / pow(n12, 2) + (2 * pow(x11, 2) * pow
        (y41, 2)) / pow(n12, 4) + 2 * pow(x11, 2) * pow(y41, 2);
4622    c2 = c2 + (8 * pow(y11, 3) * y41) / pow(n12, 2) - (4 * pow(y11, 3) * y41) / pow
        (n12, 4) - 4 * pow(y11, 3) * y41 + (8 * pow(x41, 2) * y11 * y41) / pow(n12, 2);
4623    c2 = c2 - (4 * pow(x41, 2) * y11 * y41) / pow(n12, 4) - 4 * pow(x41, 2) * y11 *
        y41 - (16 * x11 * x41 * y11 * y41) / pow(n12, 2);
4624    c2 = c2 + (8 * x11 * x41 * y11 * y41) / pow(n12, 4) + 8 * x11 * x41 * y11 *
        y41 + (8 * pow(x11, 2) * y11 * y41) / pow(n12, 2);
4625    c2 = c2 - (4 * pow(x11, 2) * y11 * y41) / pow(n12, 4) - 4 * pow(x11, 2) * y11 *
        y41 - (2 * pow(y11, 4)) / pow(n12, 2) + pow(y11, 4) / pow(n12, 4) + pow(y11,
        4);
4626    c2 = c2 - (4 * pow(x41, 2) * pow(y11, 2)) / pow(n12, 2) + (2 * pow(x41, 2) * pow
        (y11, 2)) / pow(n12, 4) + 2 * pow(x41, 2) * pow(y11, 2) + (8 * x11 * x41 * pow
        (y11, 2)) / pow(n12, 2);
4627    c2 = c2 - (4 * x11 * x41 * pow(y11, 2)) / pow(n12, 4) - 4 * x11 * x41 * pow(y11,
        2) - (4 * pow(x11, 2) * pow(y11, 2)) / pow(n12, 2) + (2 * pow(x11, 2) * pow
        (y11, 2)) / pow(n12, 4);
4628    c2 = c2 + 2 * pow(x11, 2) * pow(y11, 2) - (2 * pow(x41, 4)) / pow(n12, 2) + pow
        (x41, 4) / pow(n12, 4) + pow(x41, 4) + (8 * x11 * pow(x41, 3)) / pow(n12, 2);
4629    c2 = c2 - (4 * x11 * pow(x41, 3)) / pow(n12, 4) - 4 * x11 * pow(x41, 3) - (12 *
        pow(x11, 2) * pow(x41, 2)) / pow(n12, 2) + (6 * pow(x11, 2) * pow(x41, 2)) /
        pow(n12, 4);
4630    c2 = c2 + 6 * pow(x11, 2) * pow(x41, 2) + (8 * pow(x11, 3) * x41) / pow(n12,
        2) - (4 * pow(x11, 3) * x41) / pow(n12, 4) - 4 * pow(x11, 3) * x41 - (2 * pow
        (x11, 4)) / pow(n12, 2) + pow(x11, 4) / pow(n12, 4) + pow(x11, 4);
4631
4632    c1 = (-2 * pow(z41, 4)) / pow(n12, 2) + (6 * pow(z41, 4)) / pow(n12, 4) - (4 *
        pow(z41, 4)) / pow(n12, 6) - (2 * pow(z11, 2) * pow(z41, 2)) / pow(n12, 4);
4633    c1 = c1 + (2 * pow(z11, 2) * pow(z41, 2)) / pow(n12, 6) - (2 * pow(y41, 2) * pow
        (z41, 2)) / pow(n12, 2) + (4 * pow(y41, 2) * pow(z41, 2)) / pow(n12, 4) - (2 *
        pow(y41, 2) * pow(z41, 2)) / pow(n12, 6);
4634    c1 = c1 + (4 * y11 * y41 * pow(z41, 2)) / pow(n12, 2) - (8 * y11 * y41 * pow(
        z41, 2)) / pow(n12, 4) + (4 * y11 * y41 * pow(z41, 2)) / pow(n12, 6) - (2 * pow
        (y11, 2) * pow(z41, 2)) / pow(n12, 2);
4635    c1 = c1 + (4 * pow(y11, 2) * pow(z41, 2)) / pow(n12, 4) - (2 * pow(y11, 2) * pow
        (z41, 2)) / pow(n12, 6) - (2 * pow(x41, 2) * pow(z41, 2)) / pow(n12, 2) + (4 *
        pow(x41, 2) * pow(z41, 2)) / pow(n12, 4);
4636    c1 = c1 - (2 * pow(x41, 2) * pow(z41, 2)) / pow(n12, 6) + (4 * x11 * x41 * pow
        (z41, 2)) / pow(n12, 2) - (8 * x11 * x41 * pow(z41, 2)) / pow(n12, 4) + (4 *
        x11 * x41 * pow(z41, 2)) / pow(n12, 6);
4637    c1 = c1 - (2 * pow(x11, 2) * pow(z41, 2)) / pow(n12, 2) + (4 * pow(x11, 2) * pow
        (z41, 2)) / pow(n12, 4) - (2 * pow(x11, 2) * pow(z41, 2)) / pow(n12, 6);
4638
4639    c0 = pow(z41, 4) / pow(n12, 4) - (2 * pow(z41, 4)) / pow(n12, 6) + pow(z41, 4) /
        pow(n12, 8);
4640
4641
4642    Coeff[0] = c4;
4643    Coeff[1] = c3;
4644    Coeff[2] = c2;
4645    Coeff[3] = c1;
4646    Coeff[4] = c0;
4647
4648    for (i = 0; i < 10; i++)
4649    {
4650        SolutionReal[i] = 0;
4651        SolutionImag[i] = 0;
4652    }
4653
4654

```

```

4655     response = DKA_rh(Coeff, SolutionReal, SolutionImag, 4, 0.000000001);
4656
4657     if (response == 1)
4658     {
4659         Hitpath111 = false;
4660         goto L123;
4661     }
4662
4663     for (i = 0; i < 4; i++)
4664     {
4665         if (fabs(SolutionImag[i]) < 0.0000001)
4666         {
4667             if (SolutionReal[i] > 0.0000001)
4668             {
4669                 X = sqrt(SolutionReal[i]);
4670                 if (X * X - 1 / (pow(n12, 2)) + 1 > 0)
4671                 {
4672                     b11z = sqrt(X * X - 1.0 / (pow(n12, 2)) + 1.0);
4673                     if ((0 < b11z) && (b11z < 1))
4674                     {
4675                         b11x = (x11 - x41) / (-z41 / X + z11 / b11z);
4676                         b11y = (y11 - y41) / (-z41 / X + z11 / b11z);
4677                         if (fabs(b11x * b11x + b11y * b11y + b11z * b11z - 1) <
4678                             0.000001)
4679                         {
4680                             goto L123;
4681                         }
4682                     }
4683                 }
4684             }
4685         }
4686     }
4687
4688 L123:
4689 ;
4690
4691 }
4692
4693 int DKA_rh(double* Coeff, double* zr, double* zi, int n_dim, double deltax)
4694 {
4695     double xkjr, xkji;
4696     double fxr, fxi;
4697     double x_olldr, x_olddi;
4698     double adjr, adjdi;
4699     double th, rn, r0, max_fx;
4700     int j, k, m;
4701     int max_m;
4702     double tempr, tempi;
4703     double Pi;
4704
4705     max_m = 100;
4706     Pi = 3.14159265358979;
4707
4708     for (j = 1; j <= n_dim; j++)
4709     {
4710         Coeff[j] = Coeff[j] / Coeff[0];
4711     }
4712
4713     r0 = 0;
4714
4715     for (j = 2; j <= n_dim; j++)
4716     {
4717         rn = (double)n_dim * pow(fabs(Coeff[j]), (1 / (double)j));

```

```

4718     if (rn > r0)
4719     {
4720         r0 = rn;
4721     }
4722 }
4723
4724 for (j = 2; j <= n_dim; j++)
4725 {
4726     rn = (double)n_dim * pow(fabs(Coeff[j]), (1 / (double)j));
4727     if (rn > r0)
4728     {
4729         r0 = rn;
4730     }
4731 }
4732 for (j = 0; j <= n_dim - 1; j++)
4733 {
4734     th = (double)2 * Pi * (double)j / (double)n_dim + Pi / (2 * (double)n_dim);
4735     zr[j] = r0 * cos(th);
4736     zi[j] = r0 * sin(th);
4737 }
4738 for (m = 0; m <= max_m; m++)
4739 {
4740     max_fx = 0;
4741     for (j = 0; j <= n_dim - 1; j++)
4742     {
4743         xkjr = 1.0;
4744         xkji = 0.0;
4745         fxr = 1.0;
4746         fxi = 0.0;
4747         x_olldr = zr[j];
4748         x_oldi = zi[j];
4749         for (k = 0; k <= n_dim - 1; k++)
4750         {
4751             tempr = fxr * x_olldr - fxi * x_oldi + Coeff[k + 1];
4752             tempi = fxi * x_olldr + fxr * x_oldi;
4753             fxr = tempr;
4754             fxi = tempi;
4755             if (k != j)
4756             {
4757                 tempr = xkjr * (x_olldr - zr[k]) - xkji * (x_oldi - zi[k]);
4758                 tempi = xkji * (x_olldr - zr[k]) + xkjr * (x_oldi - zi[k]);
4759                 xkjr = tempr;
4760                 xkji = tempi;
4761             }
4762         }
4763         adjr = (fxr * xkjr + fxi * xkji) / (pow(xkjr, 2) + pow(xkji, 2));
4764         adji = (-fxr * xkji + fxi * xkjr) / (pow(xkjr, 2) + pow(xkji, 2));
4765         zr[j] = x_olldr - adjr;
4766         zi[j] = x_oldi - adji;
4767         if (sqrt(pow(fxr, 2) + pow(fxi, 2)) > max_fx)
4768         {
4769             max_fx = sqrt(pow(fxr, 2) + pow(fxi, 2));
4770         }
4771     }
4772     if (max_fx < deltax)
4773     {
4774         goto L111;
4775     }
4776 }
4777 goto L333;
4778
4779 L111:
4780 return 0;
4781 goto L222;

```

```
4782
4783 L333:
4784     return 1;
4785     goto L222;
4786
4787 L222:
4788     ;
4789
4790 }
4791
4792
4793 double* alloc_vector(long size)
4794 {
4795     double* v;
4796     if ((v = new double[size]) == NULL)
4797     {
4798         return 0;
4799     }
4800     for (long i = 0; i < size; i++)
4801     {
4802         v[i] = 0.0;
4803     }
4804     return (v);
4805 }
4806
4807 int* alloc_boolvector(long size)
4808 {
4809     int* v;
4810     if ((v = new int[size]) == NULL)
4811     {
4812         return 0;
4813     }
4814     return (v);
4815 }
4816
4817 WORD* alloc_WORDvector(long size)
4818 {
4819     WORD* v;
4820     if ((v = new WORD[size]) == NULL)
4821     {
4822         return 0;
4823     }
4824     return (v);
4825 }
4826
4827 void free_matrix(double** mtrx, long row)
4828 {
4829     long i;
4830     for (i = 0; i < row; i++) delete[] mtrx[i];
4831     delete[] mtrx;
4832 }
4833
4834
4835 double** alloc_matrix(long row, long column)
4836 {
4837     long i;
4838     double** mtrx;
4839     if ((mtrx = new double* [row]) == NULL)
4840     {
4841         return 0;
4842     }
4843     for (i = 0; i < row; i++)
4844         if ((mtrx[i] = new double[column]) == NULL)
4845         {
```

```
4846         return 0;
4847     }
4848     return mtrx;
4849 }
4850
4851 bool** alloc_boolboolmatrix(long row, long column)
4852 {
4853     long i;
4854     bool** mtrx;
4855     if ((mtrx = new bool* [row]) == NULL)
4856     {
4857         return 0;
4858     }
4859     for (i = 0; i < row; i++)
4860         if ((mtrx[i] = new bool[column]) == NULL)
4861         {
4862             return 0;
4863         }
4864     return mtrx;
4865 }
4866
4867 int** alloc_boolmatrix(long row, long column)
4868 {
4869     long i;
4870     int** mtrx;
4871     if ((mtrx = new int* [row]) == NULL)
4872     {
4873         return 0;
4874     }
4875     for (i = 0; i < row; i++)
4876         if ((mtrx[i] = new int[column]) == NULL)
4877         {
4878             return 0;
4879         }
4880     return mtrx;
4881 }
4882
4883 void free_boolmatrix(int** mtrx, long row)
4884 {
4885     long i;
4886     for (i = 0; i < row; i++) delete[] mtrx[i];
4887     delete[] mtrx;
4888 }
4889
4890 void free_boolboolmatrix(bool** mtrx, long row)
4891 {
4892     long i;
4893     for (i = 0; i < row; i++) delete[] mtrx[i];
4894     delete[] mtrx;
4895 }
4896
4897 int* alloc_intvector(long size)
4898 {
4899     int* v;
4900     if ((v = new int[size]) == NULL)
4901     {
4902         return 0;
4903     }
4904     for (long i = 0; i < size; i++)
4905     {
4906         v[i] = 0;
4907     }
4908     return (v);
4909 }
```

```
4910
4911 complex <double>* alloc_cvector(long size)
4912 {
4913     complex <double>* v;
4914     if ((v = new complex <double>[size]) == NULL)
4915     {
4916         return 0;
4917     }
4918
4919     return (v);
4920 }
4921
```