```c
//  Start of PMLcalc.h
#include "header_macro.h"
#include "constant.h"
void output_PML(const int l2, const char Yee[], const char exec[], const char PMLfile[], const int L[5],
                double u0[2*L[0]+1], double u1[2*L[1]+1],
                const int oriPML[L[4]],
                const double startPML[L[4]], const double thickPML[L[4]],
                const double sqRef[L[4]], const int powerPML[L[4]], double omega);
void input_u(const char u0u1file[], const int L[5], double u0[2*L[0]+1], double u1[2*L[1]+1]);
void input_PML(const char Yee[], const char PMLfile[],
                const int N_pml, int oriPML[N_pml],
                double startPML[N_pml], double thickPML[N_pml],
                double sqRef[N_pml], int powerPML[N_pml]);
void input_L(const char Yee[], const char PMLfile[], int L[5], double *omega);
void input_filename(FILE *fp_i1, char Yee[], char PMLfile[], char u0u1file[]);
//  End of PMLcalc.h
//=========================================================================
//   main ---- input_filename, input_file1, get_RealNum05
//=========================================================================
int main(int argc, char **argv)
{
    time_t timer_ini = time(0);    fprintf(stderr,"# Start time of PML setting = %s¥n", ctime(&timer_ini));
    if(argc != 2) {fprintf(stderr,"error: number of files ¥n"); exit(EXIT_FAILURE);}
    else if(strncmp(argv[1], "-v", 2) == 0 || strcmp(argv[1], "--version") == 0 ) {
        fprintf(stderr,"The '%s' creates perfectly matched layer (PML).¥n", argv[0]);
        fprintf(stderr,"Version 19.09.10 is compiled at %s on %s.¥n C-version   : %ld¥n", __TIME__, __DATE__, __STDC_VERSION__);
        fprintf(stderr," Source code : '%s'¥n Author      : Tatsuya Usuki¥n URL         : http://www.smatran.org¥n",   FILE  );
        fprintf(stderr," References  : 'Equation (6.6) in Discretization by Yee's lattice' as 'Formulation.pdf' on Aug 25, 2019;¥n");
        fprintf(stderr,"There is NO warranty.¥n");
        exit(EXIT_SUCCESS);//normal end
    }
//------- begin reading file names and parameters -------
    FILE *fp_i1;
    fp_i1 = fopen(argv[1],"r");
    if (fp_i1 == NULL){    fprintf(stderr,"open error!: open input-file1!¥n"); exit(EXIT_FAILURE);}
    fprintf(stderr,"The 1st input file: %s¥n",argv[1]);
    char Yee[BUFSIZE], PMLfile[BUFSIZE], u0u1file[BUFSIZE];
    input_filename(fp_i1, Yee, PMLfile, u0u1file);
    fprintf(stderr,"Prefix: %s, PMLfile: %s¥n", Yee, PMLfile);
    if(fclose(fp_i1) != 0) {    fprintf(stderr,"fclose error after input file!¥n"); exit(EXIT_FAILURE);}
    int L[5];//L[2] = bottom + scatterer + top, L[3] = outer number, L[4] = PML number
    double omega = 0.;
    input_L(Yee, PMLfile, L, &omega);
    fprintf(stderr,"L0 = %d, L1 = %d, Lbst = %d, Louter = %d, PML number = %d ¥n", L[0], L[1], L[2], L[3], L[4]);
//-------  end reading file names and parameters  -------
    {
        int *oriPML;    oriPML = malloc(sizeof(int)*L[4]);
        double *startPML;  startPML = malloc(sizeof(double)*L[4]);
        double *thickPML;  thickPML = malloc(sizeof(double)*L[4]);
        double *sqRef; sqRef = malloc(sizeof(double)*L[4]);
```

```
51          int *powerPML;  powerPML = malloc(sizeof(int)*L[4]);
52          input_PML(Yee, PMLfile, L[4], oriPML, startPML, thickPML, sqRef, powerPML);
53          {
54              double *u0;    u0 = malloc(sizeof(double)*(2*L[0]+1));
55              double *u1;    u1 = malloc(sizeof(double)*(2*L[1]+1));
56              input_u(u0u1file, L, u0, u1);
57                  for(int l2 = 0 ; l2 < L[2] ; l2++){
58                      output_PML(l2, Yee, argv[0], PMLfile, L, u0, u1, oriPML,
                        startPML, thickPML, sqRef, powerPML, omega);
59                  }
60              SAFEFREE(u0);   SAFEFREE(u1);
61          }
62          SAFEFREE(oriPML);   SAFEFREE(startPML); SAFEFREE(thickPML);
            SAFEFREE(sqRef);    SAFEFREE(powerPML);
63      }
64  }
65  //=========================================================================
66  //  output_PML  ----  matdata_file, set_header, out_flag, calc_sigma,
67  //                    sqDt_CFL           Last updated on Sep 06, 2019.
68  //=========================================================================
69  void calc_sigma(FILE *fp_i, FILE *fp_o, const int L[5],
70                  const double u0[2*L[0]+1], const double u1[2*L[1]+1], const
                    double u2[2],
71                  const int oriPML[L[4]],
72                  const double startPML[L[4]], const double thickPML[L[4]],
73                  const double sqRef[L[4]], const int powerPML[L[4]], const int
                    out_region, const double omega);
74  int out_flag(const int Ltot, const int Lout, const int istep);
75  void set_header(FILE *fp_i, double u2[2], FILE *fp_o, const int N_pml, const
    char *exec, const char PMLfile[], const double sqDt);
76  double sqDt_CFL(FILE *fp_i, const int L[5]);
77  void matdata_file(const char f_prefix[], const char add_name[], const int
    Ltot, const int Lout, const int istep, char data name[]);
78  void output PML(const int l2, const char Yee[], const char exec[], const
    char PMLfile[], const int L[5],
79                  double u0[2*L[0]+1], double u1[2*L[1]+1],
80                  const int oriPML[L[4]],
81                  const double startPML[L[4]], const double thickPML[L[4]],
82                  const double sqRef[L[4]], const int powerPML[L[4]], double
                    omega)
83  {// fprintf(stderr,"k/k 0 = oemga = %.5e¥n", omega);
84      char data name[BUFSIZE], temp name[BUFSIZE];
85      matdata file(Yee, " Med", L[2], L[3], l2, data name);
86      fprintf(stderr,"%s¥n", data name);
87      matdata file(Yee, " PML", L[2], L[3], l2, temp name);
88
89      FILE *fp r, *fp w;
90      fp r = fopen(data name,"r");
91      fp w = fopen(temp name,"w");
92      if (fp r == NULL || fp w == NULL){fprintf(stderr,"open error for %s or %s in
        output PML¥n", data name, temp name);    exit(EXIT FAILURE);}
93      double sqDt = sqDt CFL(fp r, L);
94      double u2[2];
95      set header(fp r, u2, fp w, L[4], exec, PMLfile, sqDt);//    char
        buf[BUFSIZE];  if(fgets(buf, sizeof( buf ), fp r) != NULL) {
        fprintf(fp w,"temporary %s", buf);}
96      {
97          int out region = out flag(L[2], L[3], l2);
98          calc sigma(fp r, fp w, L, u0, u1, u2, oriPML, startPML, thickPML, sqRef,
            powerPML, out region, omega);
99      }
```

```
100     if(fclose(fp_r) != 0 ||fclose(fp_w) != 0 ){ fprintf(stderr,"fclose error after reading or writing data files!¥n");  exit(EXIT_FAILURE);}
101 //  else{if(remove(data_name)==0){rename(temp_name, data_name);}}
102 }
103 //=========================================================================
104 //  calc_sigma  ----  set_sigma          Last updated on Sep 09, 2019
105 //=========================================================================
106 double set_sigma(const double x,
107                  const int oriPML,
108                  const double startPML, const double thickPML,
109                  const double sqRef, const int powerPML, const double omega);
110 void calc_sigma(FILE *fp_i, FILE *fp_o, const int L[5],
111                  const double u0[2*L[0]+1], const double u1[2*L[1]+1], const double u2[2],
112                  const int oriPML[L[4]],
113                  const double startPML[L[4]], const double thickPML[L[4]],
114                  const double sqRef[L[4]], const int powerPML[L[4]], const int out_region, const double omega)
115 {
116     char buf[BUFSIZE];
117     int j_count = 0;
118     while(fgets(buf, sizeof( buf ), fp_i) != NULL) {
119         if(strncmp(buf,"# ",2) != 0){
120             int l0, l1, j2;
121             double re[3], im[3];
122             double fx00, fx11, fx22;
123             if(sscanf(buf,"%d %d %d %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf", &l0, &l1, &j2, &re[0], &im[0], &re[1], &im[1], &re[2], &im[2], &fx00, &fx11, &fx22) == 12){
124                 if(l0 < 0 || l0 >= L[0] || l1 < 0 || l1 >= L[1] || j2 < 0 || j2 >= 2){fprintf(stderr,"%d, %d, %d error in calc_sigma!¥n", l0, l1, j2);  exit(EXIT_FAILURE);}
125                 double complex ce[3];  for(int jdata = 0 ; jdata < 3 ; jdata++){ce[jdata] = re[jdata] + I*im[jdata];}
126                 double x[9];
127                 for(int jdata = 0 ; jdata < 3 ; jdata++){
128                     int j0,j1;
129                     if(jdata == 2){ j0 = 1-j2;  j1 = 1-j2;}
130                     else{  j0 = 1 - jdata; j1 = jdata;}// Note that eq.(6.6) and check header comment of data file!
131                     x[0 + 3*jdata] = u0[2*l0+j0];//x[0 + 3*0] = u0[2*l0+1]; x[0 + 3*1] = u0[2*l0+0]; x[0 + 3*2] = u0[2*l0+1-j2];
132                     x[1 + 3*jdata] = u1[2*l1+j1];//x[1 + 3*0] = u1[2*l1+0]; x[1 + 3*1] = u1[2*l1+1]; x[1 + 3*2] = u1[2*l1+1-j2];
133                     x[2 + 3*jdata] = u2[j2];//
134                 }
135                 /*fprintf(fp_o,"%d %d %d %.5e %.5e %.5e %.5e %.5e %.5e %.5e %.5e %.5e %d %.5e¥n", l0, l1, j2,
136                     x[0 + 3*0], x[1 + 3*0], x[2 + 3*0],
137                     x[0 + 3*1], x[1 + 3*1], x[2 + 3*1],
138                     x[0 + 3*2], x[1 + 3*2], x[2 + 3*2], out_region, omega); */
139                 {   double sigma[9]; for(int j = 0 ; j < 9 ; j++){sigma[j] = 0.;}
140                     for(int jdata = 0 ; jdata < 3 ; jdata++){
141                         for(int j_pml = 0 ; j_pml < L[4] ; j_pml++){
142                             int jxyz = abs(oriPML[j_pml]) - 1;
143                             if(jxyz < 0 || jxyz >= 3){fprintf(stderr,"oriPML[%d] = %d in calc_sigma!¥n", j_pml, oriPML[j_pml]);
                                exit(EXIT_FAILURE);}
144                             else if(jxyz == 2 || out_region == 0){
145                                 sigma[jxyz+3*jdata] += set_sigma(x[jxyz + 3*jdata], oriPML[j_pml], startPML[j_pml],
```

```
                                    thickPML[j_pml], sqRef[j_pml], powerPML[j_pml],
                                    omega);//20190908 edited
146                             }
147                         }
148                     }
149                     for(int jdata = 0 ; jdata < 3 ; jdata++){
150                         int jxyz;
151                         if(j2 == 1 && jdata < 2){   jxyz = 1-jdata;}else{  jxyz =
                            jdata;}// Note that eq.(6.6) and check header comment of
                            data file!
152                         ce[jdata] *= (1. + I*sigma[(jxyz+1)%3 +3*jdata])*(1. +
                            I*sigma[(jxyz+2)%3 +3*jdata])/(1. + I*sigma[(jxyz)%3
                            +3*jdata]);
153                     }
154                 }
155                 fprintf(fp_o,"%d %d %d %.20e %.20e %.20e %.20e %.20e %.20e %.20e
                    %.20e %.20e¥n", l0, l1, j2,
156                     creal(ce[0]), cimag(ce[0]),
157                     creal(ce[1]), cimag(ce[1]),
158                     creal(ce[2]), cimag(ce[2]), fx00, fx11, fx22);
159                 j_count++;
160             }
161         }
162     }
163     if(j_count != L[0]*L[1]*2){fprintf(stderr,"%d != L[0]*L[1]*2 in
        calc_sigma¥n", j_count);    exit(EXIT_FAILURE);}
164 }
165 //=======================================================================
166 //  set_sigma                              Last updated on Sep 09, 2019
167 //=======================================================================
168 double set_sigma(const double x,
169                  const int oriPML,
170                  const double startPML, const double thickPML,
171                  const double sqRef, const int powerPML, const double omega)
172 {
173     double endPML= startPML; if(oriPML > 0){   endPML += thickPML;}else
        if(oriPML < 0){   endPML -= thickPML;}
174     if((x - startPML)*(x - endPML) <= 0.){// sqRef =
        exp[-4*omega*thickPML*max sigma/(powerPML+1)], see section 3.3
175         double max sigma = (-0.25*(powerPML+1)/(omega*thickPML))*log(sqRef);
176 //      double sigma x = max sigma * pow(fabs((x - startPML)/(endPML - startPML)),
    (double) powerPML);
177         double sigma x = max sigma * pow(fabs((x - startPML)/(endPML -
            startPML)), powerPML);
178         return(sigma x);
179     }else{
180         return(0.);
181     }
182 }
183 //=======================================================================
184 //  out flag                               Last updated on Sep 06, 2019
185 //=======================================================================
186 int out flag(const int Ltot, const int Lout, const int istep)
187 {
188     int flag;
189     if(0 <= istep && istep < Lout){
190         flag = 1;// snprintf(data name,BUFSIZE*sizeof(char),"%s%sB%d.dat",
            f prefix, add name, istep);
191     }else if(Lout <= istep && istep < Ltot - Lout){
192         flag = 0;// snprintf(data name,BUFSIZE*sizeof(char),"%s%s%d.dat",
            f_prefix, add_name, istep - Lout);
```

```
193         }else if(Ltot - Lout <= istep && istep < Ltot){
194             flag = 1;// snprintf(data_name,BUFSIZE*sizeof(char),"%s%sT%d.dat",
                f_prefix, add_name, istep - (Ltot - Lout));
195         }else{fprintf(stderr,"istep = %d error in out_flag!¥n", istep);
            exit(EXIT_FAILURE);}
196         return(flag);
197     }
198     //=========================================================================
199     //  sqDt_CFL                                Last updated on Sep 10, 2019
200     //=========================================================================
201     double sqDt_CFL(FILE *fp_i, const int L[5])
202     {
203         if (fp_i == NULL){ fprintf(stderr,"open error in set_u2¥n");
            exit(EXIT_FAILURE);}
204         char buf[BUFSIZE];
205         int j_count = 0;
206         double sqDt = -1.;
207         int l0p = -1;   int l1p = -1;
208         double mul00 = -1.;     double mul11 = -1.;     double epl22 = 0.;
209         double epl11, epl00, mul22;
210         while(fgets(buf, sizeof( buf ), fp_i) != NULL) {
211             if(strncmp(buf,"# ",2) != 0){
212                 int l0, l1, j2;
213                 double re[3], im[3];
214                 double fx00, fx11, fx22;
215                 if(sscanf(buf,"%d %d %d %lf %lf %lf %lf %lf %lf %lf %lf %lf",  &l0,
                &l1, &j2, &re[0], &im[0], &re[1], &im[1], &re[2], &im[2], &fx00,
                &fx11, &fx22) == 12){
216                     if(l0 < 0 || l0 >= L[0] || l1 < 0 || l1 >= L[1] || j2 < 0 || j2
                    >= 2){fprintf(stderr,"%d, %d, %d error in sqDt_CFL!¥n", l0, l1,
                    j2);    exit(EXIT_FAILURE);}
217                     j_count++;
218                 }
219                 if(j2 == 0){
220                     mul00 = re[0]*fx00; mul11 = re[1]*fx11; epl22 = re[2]*fx22;
221                     l0p = l0; l1p = l1;
222                 }else{
223                     epl11 = re[0]*fx00; epl00 = re[1]*fx11; mul22 = re[2]*fx22;
224                     double rdummy =
                    epl00*epl11*epl22*mul00*mul11*mul22/((epl00+epl11+epl22)*(mul00+mul1
                    1+mul22));
225                     if(rdummy > 0.){
226                         if(sqDt < 0. || sqDt > rdummy){sqDt = rdummy;}
227                     }else{fprintf(stderr,"rdummy <= 0 at  (%d, %d) error in
                    sqDt CFL!¥n", l0, l1); exit(EXIT FAILURE);}
228                     if(l0 != l0p || l1 != l1p || j2 != 1){fprintf(stderr,"(%d, %d) =
                    (%d, %d) or j2 = %d error in sqDt CFL!¥n", l0, l1, l0p, l1p, j2);
                    exit(EXIT FAILURE);}
229                 }
230             }
231         }
232         if(j count != L[0]*L[1]*2){fprintf(stderr,"%d != L[0]*L[1]*2 in sqDt CFL¥n",
            j count);  exit(EXIT FAILURE);}
233         rewind(fp i);
234         return(sqrt(sqDt));
235     }
236     //=========================================================================
237     //  set header                              Last updated on Sep 10, 2019
238     //=========================================================================
239     void set header(FILE *fp i, double u2[2], FILE *fp o, const int N pml, const
        char *exec, const char PMLfile[], const double sqDt)
```

```
240 {
241     if (fp_i == NULL){ fprintf(stderr,"open error in set_u2¥n");
      exit(EXIT_FAILURE);}
242     char buf[BUFSIZE];
243     while(fgets(buf, sizeof( buf ), fp_i) != NULL) {
244         if(strncmp(buf,"# k_0 * u_2",11) == 0){
245             if(sscanf(buf,"%*[^=] %*[=] %lf %*[;] %lf", &u2[0], &u2[1]) == 2){
246                 fprintf(fp_o,"%s", buf);
247             }
248             else{fprintf(stderr,"read error1 in set_u2¥n");exit(EXIT_FAILURE);}
249         }else if(strncmp(buf,"# l <",5) == 0){
250             char dummy_c[BUFSIZE];
251             if(sscanf(buf,"%[^¥n]", dummy_c) == 1){
252                 time_t timer_f = time(0);
253                 snprintf(buf, sizeof(buf), "%s PML num = %d, min Dt^2 = %.20e¥n#
                  PML and CFL data were added from ' %s ' by ' %s ' on %s", dummy_c,
                  N_pml, sqDt, PMLfile, exec, ctime(&timer_f));
254                 fprintf(fp_o,"%s", buf);
255             }else{fprintf(stderr,"read error2 in set_u2¥n");   exit(EXIT_FAILURE);}
256         }else{
257             if(strncmp(buf,"# ",2) == 0){fprintf(fp_o,"%s", buf);}
258             else{goto Nextstep;}
259         }
260     }
261     Nextstep:;
262     rewind(fp_i);
263 //  if(fgets(buf, sizeof( buf ), fp_i) != NULL) {   fprintf(fp_o,"%s", buf);}
264 }
265 //=====================================================================
266 //  matdata_file                          Last updated on Aug 29, 2019
267 //=====================================================================
268 void matdata_file(const char f_prefix[], const char add_name[], const int
    Ltot, const int Lout, const int istep, char data name[])
269 {
270     if(0 <= istep && istep < Lout){
271         snprintf(data name,BUFSIZE*sizeof(char),"%s%sB%d.dat", f prefix,
          add name, istep);//
          https://www.ipa.go.jp/security/awareness/vendor/programmingv1/b06 02.html
272     }else if(Lout <= istep && istep < Ltot - Lout){
273         snprintf(data name,BUFSIZE*sizeof(char),"%s%s%d.dat", f prefix,
          add name, istep - Lout);
274     }else if(Ltot - Lout <= istep && istep < Ltot){
275         snprintf(data name,BUFSIZE*sizeof(char),"%s%sT%d.dat", f prefix,
          add name, istep - (Ltot - Lout));
276     }else{fprintf(stderr,"istep = %d error!¥n", istep);exit(EXIT FAILURE);}
277 }
278 //=====================================================================
279 //  input u                               Last updated on Sep 05, 2019.
280 //=====================================================================
281 void input u(const char u0u1file[], const int L[5], double u0[2*L[0]+1],
    double u1[2*L[1]+1])
282 {
283     char buf[BUFSIZE];
284     FILE *fp i;
285     fp i = fopen(u0u1file,"r");
286     if (fp i == NULL){ fprintf(stderr,"open error for %s¥n", u0u1file);
      exit(EXIT FAILURE);}
287     int j count = -(2*L[0]+1 + 2*L[1]+1);
288     while(fgets(buf, sizeof( buf ), fp i) != NULL && j count < 0) {
289         if(strncmp(buf,"# xi0",5) == 0){
290             for(int j0 = 0 ; j0 < 2*L[0]+1 ; j0++){
```

```
291                     if(fgets(buf, sizeof( buf ), fp_i) &&
292                     sscanf(buf,"%*[^,] %*[,] %lf", &u0[j0]) == 1){j_count++;}
293                 }
294             }else if(strncmp(buf,"# xi1",5) == 0){
295                 for(int j1 = 0 ; j1 < 2*L[1]+1 ; j1++){
296                     if(fgets(buf, sizeof( buf ), fp_i) &&
297                     sscanf(buf,"%*[^,] %*[,] %lf", &u1[j1]) == 1){j_count++;}
298                 }
299             }
300         }
301         if(j_count != 0){   fprintf(stderr,"u0 and u1 can not be read @
        input_u!¥n");   exit(EXIT_FAILURE);}
302     }
303     //=====================================================================
304     //  input_PML  ---- input_orientation, input_start, input_sqRef
305     //                                      Last updated on Sep 05, 2019.
306     //=====================================================================
307     int input_orientation(const char buf[], const int n_pml0, int oriPML[]);
308     int input_start(const char buf[], const double k_0, const int n_pml0,
        double startPML[], double thickPML[]);
309     int input_sqRef(const char buf[], const int n_pml0, double sqRef[], int
        powerPML[]);
310     void rm_space( char *A );
311     void rm_comma( char *A );
312     void input_PML(const char Yee[], const char PMLfile[],
313                    const int N_pml, int oriPML[N_pml],
314                    double startPML[N_pml], double thickPML[N_pml],
315                    double sqRef[N_pml], int powerPML[N_pml])
316     {
317         char input_name[BUFSIZE], buf[BUFSIZE];
318         snprintf(input_name,sizeof(input_name),"%s_Med0.dat", Yee);
319         FILE *fp_i;
320         fp_i = fopen(input_name,"r");
321         if (fp_i == NULL){ fprintf(stderr,"open error for %s¥n", input_name);
        exit(EXIT_FAILURE);}
322         int j_count = -1;
323         double k_0;
324         while(fgets(buf, sizeof( buf ), fp_i) != NULL && j_count < 0) {
325             if(strncmp(buf,"# k_0",5) == 0 && sscanf(buf,"%*[^=] %*[=] %*[^=] %*[=]
            %lf", &k_0) == 1){fprintf(stderr,"k_0 = %.5e¥n", k_0);  j_count++;}
326         }
327         if(fclose(fp_i) != 0){  fprintf(stderr,"fclose error1 in input_PML!¥n");
        exit(EXIT_FAILURE);}
328         else if(j_count < 0){  fprintf(stderr,"k_0 can not be read in
        input_PML!¥n");  exit(EXIT_FAILURE);}
329     // (int) 0 <= N_pml <= 6    : total number of structures.
330     // (int) oriPML[N_pml]      : '+x' == +1, '+y' == +2, '+z' == +3,
331     //                            '-x' == -1, '-y' == -2, '-z' == -3.
332     // (double) startPML[N_pml] : start point of the PML [k_0^-1],
333     // (double) thickPML[N_pml] : thickness of the PML [k_0^-1],
334     // (double) sqRef[N_pml]    : reflectivity of the PML
335     // (int) powerPML[N_pml]    : index of depth dependence for the PML
336         fp_i = fopen(PMLfile,"r");
337         if (fp_i == NULL){ fprintf(stderr,"open error for %s¥n", input_name);
        exit(EXIT_FAILURE);}
338         char command[16];  // buffer for fgets
339         int n_pml0 = 0;
340         for(int J0 = 0 ; J0 < N_pml ; J0++) {
341             while(fgets(buf, sizeof( buf ), fp_i) != NULL) {
342                 rm_space(buf);
343                 if(strncmp(buf, "#", 1) != 0 && sscanf(buf,"%s", command) != EOF){
```

```
344                     rm_comma(command);
345                     if (strcmp(command, "begin") == 0) {
346                         int ori_count = 0;// Initialization
347                         int start_count = 0;
348                         int sqRef_count = 0;// End of initialization
349                         while(fgets(buf, sizeof( buf ), fp_i) != NULL) {
350                             rm_space(buf);
351                             if(strncmp(buf, "#", 1) != 0 && sscanf(buf,"%s", command)
                            != EOF){
352                                 rm_comma(command);
353                                 if (strncmp(command, "orientation", 11) == 0) {
354                                     ori_count += input_orientation(buf, n_pml0, oriPML);
355                                 }else if (strncmp(command, "start", 5) == 0) {
356                                     start_count += input_start(buf, k_0, n_pml0,
                                    startPML, thickPML);
357                                 }else if (strncmp(command, "squareRef", 9) == 0) {
358                                     sqRef_count += input_sqRef(buf, n_pml0, sqRef,
                                    powerPML);
359                                 }else if (strcmp(command, "end") == 0) {
360                                     if(ori_count != 1 || start_count != 1 ||
                                    sqRef_count != 1){
361                                         fprintf(stderr,"end error @ input_PML!:
                                        ori_count = %d, start_count = %d, sqRef_count
                                        = %d¥n",
362                                             ori_count, start_count, sqRef_count);
363                                         exit(EXIT_FAILURE);
364                                     }
365                                     else{
366                                         n_pml0 += 1;
367                                         if(n_pml0 > N_pml){
368                                             fprintf(stderr,"Error @ input_PML!: n_pml0
                                            > N_pml, n_pml0 = %d,", n_pml0);
369                                             exit(EXIT FAILURE);
370                                         }
371                                         goto NEXT step;
372                                     }
373                                 }
374                             }
375                         }
376                     }
377                 }
378             }
379             NEXT step:;
380         }
381         if(n pml0 != N pml){
382             fprintf(stderr,"Error @ input file3!: n pml0 != N pml, n pml0 = %d,",
                n pml0);
383             exit(1);
384         }else if(fclose(fp i) != 0){   fprintf(stderr,"fclose error2 in
            input PML!¥n");    exit(EXIT FAILURE);}
385         for(int J0 = 0 ; J0 < N pml ; J0++) {
386             if(oriPML[J0] == 0 || abs(oriPML[J0]) > 3){ fprintf(stderr,"oriPML[%d] ==
                oriPML[%d]¥n", J0, oriPML[J0]);   exit(EXIT FAILURE);}
387             for(int I0 = J0+1 ; I0 < N pml ; I0++) {
388                 if(oriPML[J0] == oriPML[I0]){   fprintf(stderr,"oriPML[%d] ==
                    oriPML[%d] error!¥n", J0, I0);    exit(EXIT FAILURE);}
389             }
390         }
391     }
392     //=====================================================================
393     //  input_orientation                    Last updated on Sep 05, 2019.
```

```
394 //=======================================================================
395 int input_orientation(const char buf[], const int n_pml0, int oriPML[])
396 {
397     char para[BUFSIZE];
398     if(sscanf(buf,"%*[^=] %*[=] %s", para) == 1){
399         if(strncmp(para,"+z",2) == 0 ){
400             oriPML[n_pml0] = 3;
401         }else if(strncmp(para,"+y",2) == 0 ){
402             oriPML[n_pml0] = 2;
403         }else if(strncmp(para,"+x",2) == 0 ){
404             oriPML[n_pml0] = 1;
405         }else if(strncmp(para,"-x",2) == 0 ){
406             oriPML[n_pml0] = -1;
407         }else if(strncmp(para,"-y",2) == 0 ){
408             oriPML[n_pml0] = -2;
409         }else if(strncmp(para,"-z",2) == 0 ){
410             oriPML[n_pml0] = -3;
411         }else{
412             oriPML[n_pml0] = 0;
413         }
414         fprintf(stderr,"oriPML[%d]=%d¥n", n_pml0, oriPML[n_pml0]);
415         return(1);
416     }else{
417         return(0);
418     }
419 }
420 //=======================================================================
421 //  input_start  ----  ScaleUnit         Last updated on Sep 05, 2019.
422 //=======================================================================
423 double ScaleUnit(char x[]);// unit variation: km, m, cm, mm, micron, um, nm,
    deg, degree, rad, radian.
424 int input_start(const char buf[], const double k_0, const int n_pml0,
    double startPML[], double thickPML[]){
425     char A[BUFSIZE], B[BUFSIZE];
426     if(sscanf(buf,"%*[^=] %*[=] %lf %s %*[^=] %*[=] %lf %s ",
427         &startPML[n_pml0],A, &thickPML[n_pml0],B) == 4){
428         startPML[n_pml0] *= ScaleUnit(A)*k_0;
429         thickPML[n_pml0] *= ScaleUnit(B)*k_0;
430         fprintf(stderr,"startPML[%d]=%.5e, thickPML[%d]=%.5e¥n", n_pml0,
            startPML[n_pml0], n_pml0, thickPML[n_pml0]);
431         return(1);
432     }else{
433 //      fprintf(stderr,"input start error! `%s'¥n", buf);   exit(EXIT_FAILURE);
434         return(0);
435     }
436 }
437 //=======================================================================
438 //  input sqRef                        Last updated on Sep 05, 2019.
439 //=======================================================================
440 int input_sqRef(const char buf[], const int n_pml0, double sqRef[], int
    powerPML[])
441 {
442     if(sscanf(buf,"%*[^=] %*[=] %lf %*[^=] %*[=] %d",
443         &sqRef[n_pml0], &powerPML[n_pml0]) == 2){
444         fprintf(stderr,"sqRef[%d]=%.5e, powerPML[%d]=%d¥n", n_pml0, sqRef[n_pml0],
            n_pml0, powerPML[n_pml0]);
445         return(1);
446     }else{
447 //      fprintf(stderr,"input start error! `%s'¥n", buf);   exit(EXIT_FAILURE);
448         return(0);
449     }
```

```
450 }
451 //=======================================================================
452 //  input_L ---- input_N_str
453 //                                        Last updated on Sep 05, 2019.
454 //=======================================================================
455 int input_N_str(FILE *fp_i3);
456 void input_L(const char Yee[], const char PMLfile[], int L[5], double *omega)
457 {
458     char input_name[BUFSIZE], buf[BUFSIZE];
459     snprintf(input_name,sizeof(input_name),"%s_Med0.dat", Yee);
460     FILE *fp_i;
461     fp_i = fopen(input_name,"r");
462     if (fp_i == NULL){ fprintf(stderr,"open error for %s¥n", input_name);
    exit(EXIT_FAILURE);}
463     int j_count = -3;
464     double wavelength, k_0;
465     while(fgets(buf, sizeof( buf ), fp_i) != NULL && j_count < 0) {
466 //      if(strncmp(buf,"# info",6) == 0 && sscanf(buf,"%*[^=] %*[=] %*[^=] %*[=]
    %s", dummy) == 1){rm_comma(dummy); j_count++;}
467 //      else if(sscanf(buf,"%*[^<] %*[<] %d %*[^<] %*[<] %d %*[^<] %*[<] %d %*[^<]
    %*[<] %*[^=] %*[=] %d", &L[0], &L[1], &L[2], &L[3]) == 4){j_count++;}
468         if(sscanf(buf,"%*[^<] %*[<] %d %*[^<] %*[<] %d %*[^<] %*[<] %d %*[^<]
        %*[<] %*[^=] %*[=] %d", &L[0], &L[1], &L[2], &L[3]) == 4){j_count++;}//-2
469         else if(strncmp(buf,"# wavelength",12) == 0 && sscanf(buf,"%*[^=] %*[=]
        %lf", &wavelength) == 1){j_count++;}//-1
470         else if(strncmp(buf,"# k_0",5) == 0 && sscanf(buf,"%*[^=] %*[=] %*[^=]
        %*[=] %lf", &k_0) == 1){j_count++;}//0
471     }
472     if(fclose(fp_i) != 0){  fprintf(stderr,"fclose error after reading a data
    file!¥n");    exit(EXIT_FAILURE);}
473     else if(j_count < 0){  fprintf(stderr,"L,M,N,N_outer can not be read at
    Scatterer @ input_file0!¥n");  exit(EXIT_FAILURE);}
474     else{  fprintf(stderr,"wavelength = %.5e, k_0 = %.5e¥n", wavelength, k_0);
    *omega = Pi2/(wavelength*k_0);}
475
476     fp_i = fopen(PMLfile,"r");
477     if (fp_i == NULL){ fprintf(stderr,"open error for %s¥n", PMLfile);
    exit(EXIT_FAILURE);}
478     L[4] = input_N_str(fp_i);
479 }
480 //=======================================================================
481 //  input filename ---- rm space, rm comma
482 //                                        Last updated on Sep 05, 2019
483 //=======================================================================
484 //void rm_space( char *A );
485 //void rm_comma( char *A );
486 void input_filename(FILE *fp_i1, char Yee[], char PMLfile[], char u0u1file[])
487 {
488     char buf[BUFSIZE]; // buffer for fgets
489     int j_count = -3;
490     while(fgets(buf, sizeof( buf ), fp_i1) != NULL && j_count < 0) {
491         rm_space(buf);
492         if(strncmp(buf, "xi2u", 4) == 0 && sscanf(buf,"%*[^=] %*[=] %s",
        u0u1file) == 1){
493             rm_comma(u0u1file);
494             j_count++;// -2
495         }else if(strncmp(buf, "Prefix", 6) == 0 && sscanf(buf,"%*[^=] %*[=] %s",
        Yee) == 1){ //Prefix = Yee
496             rm_comma(Yee);
497             j_count++;// -1
498         }else if(strncmp(buf, "PML", 3) == 0 && sscanf(buf,"%*[^=] %*[=] %s",
```

```
            PMLfile) == 1){
499             rm_comma(PMLfile);
500             j_count++;// 0
501         }
502     }
503     if(j_count != 0) {  fprintf(stderr,"Control commands can not read in
        input_filename!"); exit(EXIT_FAILURE);}
504 }
505 //=========================================================================
506 //  This program removes spaces of head in characters,
507 //                              it needs #include <ctype.h>
508 //                            Last updated on Jun 05, 2018.
509 //=========================================================================
510 void rm_space(char *A)
511 {
512 //  fprintf(stderr,"Read data before removing spaces =%s¥n", A);
513 //  A[BUFSIZE - 1] = '¥0';// A[] has to include NULL character.
514     while(isspace( A[0] ) != 0){
515         int i = 1;
516         while(A[i] != '¥0') {
517             A[i-1] = A[i];
518             i++;
519         }
520         A[i-1] = '¥0';
521     }
522 //  fprintf(stderr,"Read data after removing spaces =%s¥n", A);
523 }
524 //=========================================================================
525 //  This program removes comma of end in characters.
526 //                                          Last updated on Jul 06, 2018
527 //=========================================================================
528 void rm_comma(char *A)
529 {
530 //  A[BUFSIZE - 1] = '¥0';// A[] has to include NULL character.
531     int i = 0;
532     while(A[i] != '¥0' && i < BUFSIZE){
533         if(A[i] == ',') {
534             A[i] = '¥0';
535             goto replaced;
536         }
537         i++;
538     }
539     replaced:;
540 }
541 //=========================================================================
542 //  input N str
543 //                                          Last updated on Sep 08, 2014
544 //=========================================================================
545 int input N str(FILE *fp i3)
546 {
547     char buf[16], command[8];  // buffer for fgets, command for seeking "begin"
        and "end".
548     int n str0 = 0;
549     while(fgets(buf, sizeof( buf ), fp i3) != NULL) {
550         if(sscanf(buf,"%s", command) != EOF){
551             if (strcmp(command, "begin") == 0) {
552                 while(fgets(buf, sizeof( buf ), fp i3) != NULL) {
553                     if(sscanf(buf,"%s", command) != EOF){
554                         if ( strcmp(command, "end") == 0) {
555                             n str0 += 1;
556                             goto NEXT_step;
```

```
557                             }
558                         }
559                     }
560                 }
561             }
562             NEXT_step:;
563         }
564         return(n_str0);
565     }
566     //=============================================================================
567     //  double ScaleUnit
568     //                                          Last updated on Feb 26, 2017.
569     //=============================================================================
570     //#include <string.h>
571     double ScaleUnit(char x[])
572     {
573         double scale;
574         if (strncmp(x, "cm", 2) == 0) {
575             scale = 1e-2;
576         } else if (strncmp(x, "deg", 3) == 0) {
577             scale = Pi/180.;
578         } else if (strncmp(x, "km", 2) == 0) {
579             scale = 1e3;
580         } else if (strncmp(x, "m", 1) == 0) {
581             if (strncmp(x, "mm", 2) == 0) {
582                 scale = 1e-3;// order of m, mm, micron is important!
583             } else if (strncmp(x, "micron", 6) == 0) {
584                 scale = 1e-6;// order of m, mm, micron is important!
585             } else {
586                 scale = 1e-0;// order of m, mm, micron is important!
587             }
588         } else if (strncmp(x, "nm", 2) == 0) {
589             scale = 1e-9;
590         } else if (strncmp(x, "um", 2) == 0) {
591             scale = 1e-6;
592         } else if (strncmp(x, "rad", 3) == 0) {
593             scale = 1.;
594         } else {
595             fprintf(stderr,"ScaleUnit error!  x = %s¥n", x);
596             exit(1);
597         }
598         return(scale);
599     }
600
```