```
#pragma warning( disable: 0266 )
#pragma warning( disable: 4101 )
#pragma warning( disable: 4996 )
#pragma warning( disable: 6031 )
#pragma warning( disable: 6001 )
#pragma warning( disable: 6385 )
#pragma warning( disable: 6386 )
#pragma warning( disable: 6262 )
#pragma warning( disable: 26451 )

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <direct.h>
#include <Windows.h>
#include <tchar.h>
#include <locale>
#include <cstdlib>
using namespace std;


//! 諸定数の定義
#define PI 3.141592653
#define C 2.99792458e8
#define Epsilon0 8.8541878e-12
#define Mu0 PI*4.0e-7

//! 関数副プログラムの読み込み
// 屈折率波長微分
double dndl ( double lamda, double n_lamda, int mater );
// 屈折率濃度微分
double dndc ( double lamda, int mater );
// コア中心屈折率
double ncore ( double lamda, int mater );
// クラッド屈折率
double nclad ( double lamda, int mater );
// 第1種変形Bessel関数 In(x)
double  bessi0 (double x), bessi1 (double x);
// 第2種変形Bessel関数 Kn(x)
double  bessk0 (double x),  bessk1 (double x), bessk ( int n, double x );
// 係数行列要素計算関数
void S_matrix ( double *a, double *b, double *q, int m, int n, double v, double w,
  double D );
// 改訂コレスキー分解
void mcholesky ( double *a, double *b, double *ML, double *MD, int m, int n );
// 改訂コレスキー分解法により方程式を解く
void mcholesky_sol ( double *ML, double *MD, double *R, int m, int n );
// 逆べき乗法の初期ベクトル計算
void R0 ( double *MD, double *R, int m, int n );
// 逆べき乗法の解ベクトル規格化
void R_norm ( double *R, int n );
// 固有値計算
double Eigen ( double *R, double *a, double *b, int n, int m );
// 群遅延計算用関数
double dbdk_bunbo ( double *R, double D, double w, int m, int n );
// 群遅延計算用関数
double dbdk_bunshi ( double *R, double *q2, double D, double w, int m, int n);
// メモリ領域確保（整数ベクトル用）
int *dintvector ( int i, int j );
// メモリ領域解放（整数ベクトル用）
void free_dintvector ( int *a,  int i );
// メモリ領域確保（実数ベクトル用）
```

```
 65  double *drealvector ( int i, int j );
 66  // メモリ領域解放（実数ベクトル用）
 67  void free_drealvector ( double *a,  int i );
 68  // メモリ領域確保（マトリクス用）
 69  double **dmatrix ( int nr1, int nr2, int nl1, int nl2 );
 70  // メモリ領域解放（マトリクス用）
 71  void free_dmatrix ( double **a, int nr1, int nr2, int nl1, int nl2 );
 72  // 2次元実数配列初期化
 73  void init_realmatrix ( double **a, int nr1, int nr2, int nl1, int nl2 );
 74  // 1次元整数配列初期化
 75  void init_intvector ( int *a, int nr1, int nr2 );
 76  // 1次元実数配列初期化
 77  void init_realvector ( double *a, int nr1, int nr2 );
 78  // 畳み込み積分実行関数
 79  double* convolution ( int n1, double* P1, int n2, double* P2 );
 80  // ディレクトリ作成関数
 81  void mkdir(char dirname[]);
 82  //TODO 既存ディレクトリ削除後、ディレクトリ作成関数［不必要］
 83  //void delmkdirconfirm(char dirname[]);
 84  //!? FEMによる, VCSEL発振モードの電磁界分布の算出
 85  char inputFEM();
 86  //!? 発振モードの選定
 87  void selectOsciMode(char *directory);
 88  //!? -3dB帯域幅計算関数
 89  double sweep_g(int SingleSweep, double ginput);
 90
 91
 92  /* 1. パラメータの定義 */
 93  // m: モード次数( TE&TM ~ 1, EH ~ n+1, HE ~ n-1), l: 動径方向モード次数, n: 方位角
       モード次数
 94  // N: 動径座標rのコア内分割数, Nbeta: 伝搬定数の分割数, mater: 材料ID,
 95  // lamda: 波長, A: コア半径, g: 屈折率次数, n0: コア中心の屈折率, n1: クラッドの屈折
       率, dr: 動径座標刻み幅
 96  // k: 波数, delta: 比屈折率, NA: 開口数, aa: 動径座標規格化サイズ
 97  // v: 規格化周波数, w: 規格化伝搬定数, D: 規格化コア径
 98  // tau: 群速度, beta: 伝搬定数β, dbeta: 伝搬定数刻み幅
 99  // a: 係数行列Sの副対角要素格納配列, b: 係数行列Sの対角要素格納配列
100  // GI: コア内屈折率, q: 規格化コア内屈折率, q2: コア内屈折率分散パラメータ
101  // R: 規格化横方向電場成分, Rb: 逆べき乗法用入れ子配列
102  // ML: LDU分解係数行列のL行列副対角要素, MD: LDU分解係数行列のD行列対角要素
103  // dd, ds: 逆べき乗法における収束判別パラメータ, eig: 固有値
104  // modem: 方位角モード次数, model: 動径モード次数, modep: 主モード次数
105  // Nz: 総ステップ数, Nzout: ファイル出力基準ステップ数
106  // Li: i番目の微小区間における相対遅延ステップ数の最大値
107  // kim: i番目の微小区間における各モードの相対遅延ステップ数
108  // beta: 伝搬定数β, tau: 群遅延, taumin: 最小群遅延, taumax: 最大群遅延
109  // zmax: ファイバ長, dZ: 空間座標刻み幅, Tv: 時間刻み幅
110  // hmn: 電力結合係数, Hmmmin: H行列対角要素最小値, Hrowsum:
111  // fmax: 最大評価周波数, Nf: 評価周波数範囲の分割数, df: 評価周波数分解能 (=fmax/Nf)
112  // nmax: プロファイルループ内におけるLmaxの最大値.
113  // nstd: 相対時間の基準値 (y=0のtauimnを基準としたn=0の補正値)
114  // Dc: 相関長, sigma2: microbendingの軸揺らぎの分散
115  // wo: 摂動設定用パラメータ (0: w/o coupling, 1: w/ coupling_heterogeneity, 2: w/
       coupling_microbending)
116  // ftou: ファイル出力設定用パラメータ (0: 部分出力, 1: 全出力)
117  // matdis: 材料分散考慮パラメータ (0:無視, 1:考慮)
118  // nP: インパルス応答格納配列用確保領域
119  // A00: 入力インパルス振幅, Aw00: 入力インパルス振幅のスペクトル成分合計
120  // 行列 A-(+) (各節点におけるモード結合前後のモードパワー分布)
121  // 行列 H (各節点における伝達関数)
122  // 配列 kim (i番目の微小区間におけるモードmの相対遅延ステップ数)
123
124  //! 以下メイン関
       数 ////////////////////////////////////////////////////////////
125  int main ( void ) {
```

```
126
127     FILE  *fptr, *fgvsBW;
128     char   directory[128];
129     int    SingleSweep, rMeasure;
130     double gmin, gmax, dg, ginput, bw;   ginput = bw =0;
131
132     sprintf(directory, "%s", inputFEM);
133     selectOsciMode(directory);
134     if ( (fptr = fopen("[BW_Input_index_exponent].csv","r")) != NULL ) {
135         char ss1 [128], ss2 [128];
136          fscanf (fptr, "%[^,], %[^,], %d¥n", ss1, ss2, &SingleSweep);
137          fscanf (fptr, "%[^,], %[^,], %lf¥n", ss1, ss2, &gmin );
138          fscanf (fptr, "%[^,], %[^,], %lf¥n", ss1, ss2, &gmax );
139          fscanf (fptr, "%[^,], %[^,], %lf¥n", ss1, ss2, &dg );
140          //!? 下行は今後削除する予定
141          fscanf(fptr, "%[^,], %[^,], %d¥n", ss1, ss2, &rMeasure);   }
142     else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE ); }
143
144     if (SingleSweep == 0){
145         bw = sweep_g(SingleSweep, ginput);  //(返り値は-3dB帯域幅)
146     }
147
148     if (SingleSweep == 1){
149         if ((fgvsBW = fopen("[BW_Input_index_exponent].csv", "r")) != NULL) {
150             fprintf(fgvsBW, "index exponent g, -3dB bandwidth¥n");
151             for (int gcnt = 0; gcnt <= (gmax - gmin) / dg; gcnt++) {
152                 ginput = gmin + dg * gcnt;
153                 //! bw = sweep_g(SingleSweep, ginput);  //(返り値は-3dB帯域幅)
154                 //! fprintf(fgvsBW, "%.2lf, %lf¥n", ginput, bw);
155     }}}
156 }
157 //! メイン関数終了  ////////////////////////////////////////////////////////
158
159
160
161 //! 以下サブ関数  ////////////////////////////////////////////////////////
162 double sweep_g(int SingleSweep, double ginput){
163     //! ==========  宣言  ==========
164     FILE  *fp,*fp2, *fp3, *fp4, *fq, *fr, *fr2, *fr3, *fr4, *fr5,*fr6, *fr7, *fr8,
          *fr9, *fr10, *fs, *fs2, *fs3, *fs4;
165     int    i, j, l, m, n, x, y, nr, jmax, count = 0;
166     int    mater, Nl, N, Nclad, Nbeta, NLP, NLP0, Ntotal, Nwkb, Ptotal, myu, nyu,
          nstd, nstdmin, nstdmax, mm;
167     int    Nz, Nzout, Nf, Nfp, Ti, Li, Lmax, nmax, wo, gi, ss, fout, matdis, scc, nP,
           Mn, launch, Nxy, Nom;
168     double lamda, lamda0, lamdamin, lamdamax, lpmin, lpmax, dlp, fp0, dl, k, omega,
          A, AA, g, n0, n1, dr;
169     double delta, NA, aa, v, w, D, w0, r0, dx, dy, xx, yy, rr, Rxy, Ein, Emev, Emod,
          Amev, Amod, OffRes, OffRan;
170     double tau, beta, dbeta, bb, eps1, eps2, sum, sumcore, sumclad, Rinf, dd, ds,
          eig;
171     double deps2, Dc, sigma2, Db, dbmn, E_over,  hmn;
172     double zmax, zout, dz, Tv, Hmmmin, Hrowsum, tauminstd, nctaumax, taumax, taumin =
           0;
173     double fmin, fmax, df, fpmin, fpmax, dfp, A00, Aw00, Hw, ReHw, ImHw, me, bw, tav,
           rms, Ptot, ap, spct, Cpsum;
174     double *GI, *GC, *q, *q2, *R, *R2, *Rb, *a, *b, *ML, *MD, **Rlp, *Mtau, *Mbeta;
175     double **Amin, **Amplu, **H, *alpha, *P, *Pm, *Pg, *M, *Cp, *Pin, *Pout, *GIND,
          *OSin, *Wl, * WDin;
176     int    *model, *modem,*modep, *pdeg, *kim, *Pnum;
177     double V0, nv0, nv1, gsingle;   gsingle = 0;
178
179     //! ////////////////////////////////////////////////////////////
180     //! ==========          1-1. 初期設定          ==========
181     //! ////////////////////////////////////////////////////////////
```

```
182     /*  入力ファイルの読み込み */
183     if ( (fp = fopen("[BW_Input].csv","r")) != NULL ) {
184         char s1[128], s2[128];
185          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &mater );
186          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &gsingle);
187          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &A );
188          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &zmax );
189          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &lamda0 );
190          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &wo );
191          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &deps2 );
192          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &Dc );
193          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &sigma2 );
194          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &Db );
195          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &launch );
196          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &r0);
197          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &dx );
198          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &w0 );
199          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &OffRes);
200          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &OffRan);
201          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &matdis );
202          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &lamdamin );
203          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &lamdamax );
204          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &Nl );
205          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &scc );
206          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &lpmin );
207          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &lpmax );
208          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &dlp );
209          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &Ti );
210          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &gi );
211          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &fout );
212          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &AA );
213          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &dr );
214          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &aa );
215          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &Nbeta );
216          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &eps1 );
217          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &eps2 );
218          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &jmax );
219          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &dz );
220          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &Tv );
221          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &zout );
222          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &fmin );
223          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &fmax );
224          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &Nf );
225          fscanf ( fp, "%[^,], %[^,], %d¥n", s1, s2, &nP );
226          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &V0);
227          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &nv0);
228          fscanf ( fp, "%[^,], %[^,], %lf¥n", s1, s2, &nv1); }
229      else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE ); }
230     fclose(fp);
231
232     //! g値の上書き
233      if (SingleSweep == 0){ g = gsingle;    }
234      if (SingleSweep == 1){ g = ginput;     }
235
236
237      /* 入力パラメータの単位変換 */
238      N = (int) ( 1000.0*A / dr ); Nclad = (int)( 1000.0*( AA - A ) / dr ); // 面内動 ➲
           径方向ステップ数の換算（必ずこの位置で定義）.
239      if ( Nl%2 != 0 ) { Nl = Nl+1;} dl = ( lamdamax - lamdamin ) / (double)Nl;  // 材 ➲
           料分散評価用波長ステップ
240      fpmin = C / lpmax, fpmax = C / lpmin, Nfp = int (( lpmax - lpmin ) / dlp), dfp = ➲
           (fpmax-fpmin) / (double) Nfp; // 周波数領域の光源スペクトルの定義.
241      lamda0 = lamda0*1.0e-9; lamdamin = lamdamin*1.0e-9; lamdamax = lamdamax*1.0e-9; ➲
           dl = dl*1.0e-9; // 単位変換（m）
242      lpmin = lpmin*1.0e-9, lpmax = lpmax*1.0e-9, dlp = dlp*1.0e-9; // 単位変換（m）
```

```
243         dfp = dfp*1.0e-3; fpmin = fpmin*1.0e-3; fpmax = fpmax*1.0e-3; // 単位変換（THz）
244         A = A*1.0e-6; dr = dr*1.0e-9; AA = AA *1.0e-6; // 単位変換（m）
245         w0 = w0*1.0e-6; r0 = r0*1.0e-6, dx = dy = dx*1.0e-9; // 単位変換（m）
246         Nxy = (int) ( 5.0*w0 / dx );//とりあえず4w0の範囲
247         Dc = Dc*1.0e-9, Db = Db*1.0e-3, deps2 = deps2 *(Epsilon0)*(Epsilon0); // 単位変
              換（m），比誘電率を誘電率に変換
248         fmin = fmin*1.0e-3; fmax = fmax*1.0e-3; df = ( fmax-fmin ) / (double) Nf;  // 単
              位変換（THz(1/ps)）
249         Nz = (int) ( zmax / dz ); Nzout = (int) ( zout / dz ); // ファイバ軸方向分割数お
              よびファイル出力間隔の換算.
250         D = A / aa; // 規格化コア径の換算.
251         lamda = lamda0; Aw00 = 0.0; spct = Cpsum = 0.0; // 変数初期化
252         xx = yy = rr = Rxy = 0.0;
253         if ( matdis == 0 ) { Nl = 0; }
254
255         /* 入力データの格納 */
256         // 入射光時間波形
257         Pin = drealvector ( 0, Ti ); init_realvector ( Pin, 0, Ti );
258         if ( (fr5 = fopen ("Input_pulse_waveform.csv", "r") ) != NULL ) {
259             for ( i = 0; i < Ti; i++ ) { fscanf ( fr5, "%lf¥n", &Pin[i] ); }}
260         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE );  }
261         fclose (fr5);
262         // 入射光スペクトル
263         OSin = drealvector ( 0, Nfp ); init_realvector ( OSin, 0, Nfp );
264         Wl = drealvector ( 0, Nl+1 );  init_realvector ( Wl, 0, Nl+1 );
265
266         if ( (fr8 = fopen ("[OS_Input_source_spectrum].csv", "r") ) != NULL ) {
267             for ( i = 0; i <= Nfp; i++ ) { fscanf ( fr8, "%lf¥n", &OSin[i] ); }}//, %lf
                    &Wl[i],
268         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE );  }
269         fclose(fr8);
270         // 入射光スペクトルにおける発振モードの判別波長を入力
271         if ( (fr10 = fopen ("[OS_Wavelength_discrimination].csv", "r") ) != NULL ) {
              //TODO WDin[ ]は境界波長を示す
272             char s1[128], s2[128];
273             fscanf(fr10, "%[^,], %[^,], %d¥n", &s1, &s2, &Nom);    printf("%d¥n", Nom);
274             WDin = drealvector(0, Nom);    init_realvector(WDin, 0, Nom);
275             for ( i = 0; i < Nom; i++ ) { fscanf ( fr10, "%lf¥n", &WDin[i] );   printf
                  ("%lf¥n", WDin[i]);}}//, %lf    &Wl[i],
276         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE );  }
277         fclose(fr10);
278         // 屈折率分布（@589nm）＆ドーパント濃度分布分布
279         if ( gi == 1) {
280             GIND = drealvector ( 0, N ); init_realvector ( GIND, 0, N );
281             if ( (fr7 = fopen ("GI_profile_NaD.csv", "r") ) != NULL ) {
282                 for ( i = 0; i <= N; i++ ) { fscanf ( fr7, "%lf¥n", &GIND[i] ); }}
283             else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE );  }
284             GC = drealvector ( 0, N ); init_realvector ( GC, 0, N );
285             for ( i = 0; i <= N; i++ ) { GC[i] = ( GIND[i] - GIND[N] ) / dndc ( 589.0,
                  mater); }
286             fclose (fr7);}
287
288         //! 時間があれば以下を行いたい
289         //  時間があれば、出力ディレクトリの設定を行いたい
290         //  （なかなかデバッグできない場合は下記を作成する必要あり）
291         /*  fopen関数のwriteタイプで開くファイルは、下記の出力ファイルの設定で全てである
              ため、
292             下記をディレクトリに収めて
293             g=%.2lfを含めて関数を分け、返り値 -3dB帯域幅*でやりたい
294             編集するなら、
295             ①（SingleSweep==1）ディレクトリ・g値の2つをBW_settingなどの出力ファイル名に
                  つける（ex. BW_setting¥）
296             ②（SingleSweep==0）井上さんからもらった状態のまま出力ファイルの設定を行う*/
297
298         //! 出力ファイルの設定
```

```
299         // 評価条件一覧
300         //! sprintf(frpath, "%s/profile.csv", directory);
301         if ( ( fp2 = fopen("BW_setting.csv","w")) != NULL ) {}
302         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE ); }
303         // 屈折率プロファイル
304         if ( ( fp3 = fopen ("BW_profile.csv", "w")) != NULL ) {
305             if (gi == 0) { fprintf (fp3, "r [um],n,q,q2¥n"); } if (gi == 1) { fprintf
                (fp3, "r [um],c [wt pct],n,q,q2¥n"); }}
306         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE ); }
307         // 有限要素法計算結果
308         if ( ( fp4 = fopen ("FEM_result.csv", "w")) != NULL ) {}
309         else { printf (" U cannot open the file !¥n"); exit ( EXIT_FAILURE ); }
310         // 電力結合係数計算結果
311         if ( ( fq = fopen ( "Hmn_result.csv", "w" )) != NULL ) {
312             if ( wo == 1 ) {
313                 fprintf ( fq, "μ,m (mode μ),l (mode μ),p (mode μ),β (mode μ),ν,m
                    (mode ν),l (mode ν),p (mode ν),β (mode ν),Δp,|Δβ|,E-field
                    overlap,hμν¥n");}
314             if ( wo == 2 ) {
315                 fprintf ( fq, "μ,m (mode μ),l (mode μ),p (mode μ),β (mode μ),ν,m
                    (mode ν),l (mode ν),p (mode ν),β (mode ν),Δp,|Δβ|,hμν¥n");}}

316         else { printf ( "U cannot open the file!¥n" ); exit ( EXIT_FAILURE ); }
317         // 行列H
318         if ( ( fr = fopen ( "CPE_Hmatrix.csv", "w" ) ) != NULL ) {}
319         else { printf ( "The file cannot be opened !¥n" ); exit ( EXIT_FAILURE ); }

320         // インパルス応答波長成分 P
321         if ( ( fr2 = fopen ( "CPE_Impulse-responce.csv", "w" ) ) != NULL ) {}
322         else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
323         // 出射波形波長成分 Pout
324         if ( ( fr6 = fopen ( "CPE_Output-pulse-waveform.csv", "w" ) ) != NULL ) {}

325         else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
326         // モードパワー分布 Pm
327         if ( ( fr3 = fopen ( "CPE_Mode-power-distribution.csv", "w" ) ) != NULL ) {}

328         else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
329         // モード群パワー分布 Pg
330         if ( ( fr9 = fopen ( "CPE_Group-power-distribution.csv", "w" ) ) != NULL ) {}

331         else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
332         // 周波数応答 M
333         if ( ( fr4 = fopen ( "CPE_Frequency-respnse.csv", "w" ) ) != NULL ) {
334             fprintf ( fr4, "," ); for ( j = 0; j <= Nf; j++ ) { fprintf ( fr4, "%f,",
                (fmin*1.0e3)+(double)j*(df*1.0e3) ); } fprintf ( fr4, "¥n" ); }
335         else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
336         // 主要な結果
337         if ( ( fs = fopen ( "BW_result.csv", "w" )) != NULL ) {
338             if ( matdis == 0) { fprintf ( fs, "g value,Nlp,Nlp0,tstart [ns],length
                [m],pulse broadening [ps],-3dB bandwidth [GHz],tav[ps],Ptot,Aw00,spct
                ¥n"); }
339             if ( matdis == 1) { fprintf ( fs, "g value,length[m],pulse broadening
                [ps],-3dB bandwidth [GHz],tav[ps],Ptot,Aw00,spct¥n"); }}
340 //          else { fprintf ( fs, "g value,length[m],-3dB bandwidth [GHz],rms width
    [ps],tav[ps],Ptot,Aw00,spct¥n"); }}
341         else { printf ( "U cannot open the file!¥n" ); exit ( EXIT_FAILURE ); }
342         // インパルス応答
343         if ( ( fs3 = fopen ( "BW_Impulse-responce.csv", "w" )) != NULL ) {
344             for ( n = 0; n <= nP; n++ ) { fprintf ( fs3, "%f,", double(n)*Tv ); }
                fprintf ( fs3, "¥n" ); }
345         else { printf ( "U cannot open the file!¥n" ); exit ( EXIT_FAILURE ); }
346         // 出射波形波長成分 Pout
347         if ( ( fs4 = fopen ( "BW_Output-pulse-waveform.csv", "w" ) ) != NULL ) {}
```

```
348       else { printf ("The file cannot be opened !¥n"); exit ( EXIT_FAILURE ); }
349       // 光源スペクトル
350       if ( ( fs2 = fopen ( "BW_source-spectrum.csv", "w" )) != NULL ) {
351               for ( i = 0; i <= Nfp; i++ ) { fprintf ( fs2, "%f,%f,%f¥n", fpmin+double
                    (i)*dfp,1.0e-3*C / ( fpmin+double(i)*dfp ), OSin[Nfp-i] ); }}
352       else { printf ( "U cannot open the file!¥n" ); exit ( EXIT_FAILURE ); }
353
354       printf ( "Total spatial steps Nf: %d¥n", Nf );
355
356       //! ////////////////////////////////////////////////////////
357       //! //////////          1-2. Input FEM              ///////////
358       //! ////////////////////////////////////////////////////////
359
360       if (launch != 2) {
361           goto SkipInputFEM;  }
362
363
364
365
366
367   SkipInputFEM:
368
369
    //! ////////////////////////////////////////////////////////////////
370           //! //////////      ( 2. ~ 5. のループにおける設定の出
                力 )          ///////////
371           //! ////////////////////////////////////////////////////////////////
372           /* 変数の初期化 */
373           Aw00 = tauminstd = 0.0; nmax = nstd = nstdmin = nstdmax = Mn = 0;
374           /* 配列の記憶領域確保および初期化 */
375           if ( matdis == 1 ) { P = drealvector ( 0, nP ), init_realvector ( P, 0,
                nP ); }
376           Pnum= dintvector ( 0, Nl ); init_intvector ( Pnum, 0, Nl );
377           M = drealvector ( 0, Nf ); init_realvector ( M, 0, Nf );
378           int contct=0;
379           for ( y = 0; y <= Nl; y++ ) {
380               /* 波長指定 */
381               if ( matdis == 1 ) { lamda = lamdamax -  y*dl; }
382               /* 各種パラメタ計算 */
383               if ( gi == 0 ) { n1 = ncore ( lamda*1.0e9, mater ); n0 = nclad
                    ( lamda*1.0e9, mater ); }
384               if ( gi == 1 ) { n1 = dndc ( lamda*1.0e9, mater )*GC[0] + nclad
                    ( lamda*1.0e9, mater ); n0 = nclad ( lamda*1.0e9, mater ); }
385               delta = (n1*n1-n0*n0) / (2.0*n1*n1), NA = sqrt (n1*n1-n0*n0);
386               k = 2.0*PI / lamda; omega = 2.0*PI*C / lamda; v = k*aa*n1*sqrt
                    ( 2.0*delta );
387               Nwkb = int( (1.0/4.0)*( g / (g+2.0) )*(k*k)*(n1*n1)*delta*(A*A) );
388               /* 変数の初期化 */
389               bb = -1; dbeta=  k*( n1 - n0 ) / (double) Nbeta;
390               NLP= 0;  NLP0 = Ntotal = Ptotal = 0;
391               /* 配列の記憶領域確保および初期化 */
392               GI = drealvector ( 0, N ); init_realvector ( GI, 0, N );
393               q = drealvector ( 0, N ); init_realvector ( q, 0, N );
394               q2 = drealvector ( 0, N ); init_realvector ( q2, 0, N );
395               R= drealvector ( 0, N ); init_realvector ( R, 0, N );
396               R2 = drealvector ( 0, N ); init_realvector ( R2, 0, N );
397               Rb = drealvector ( 0, N ); init_realvector ( Rb, 0, N );
398               a = drealvector ( 0, N ); init_realvector ( a, 0, N );
399               b = drealvector ( 0, N ); init_realvector ( b, 0, N );
400               ML = drealvector ( 0, N ); init_realvector ( ML, 0, N );
401               MD = drealvector ( 0, N ); init_realvector ( MD, 0, N );
402               modem = dintvector ( 0, 2*Nwkb ); init_intvector ( modem, 0, 2*Nwkb );
403               model = dintvector ( 0, 2*Nwkb ); init_intvector ( model, 0, 2*Nwkb );
404               modep = dintvector ( 0, 2*Nwkb ); init_intvector ( model, 0, 2*Nwkb );
405               pdeg = dintvector ( 0, Nwkb ); init_intvector ( pdeg, 0, Nwkb );
406               Mbeta = drealvector ( 0, 2*Nwkb ); init_realvector ( Mbeta, 0, 2*Nwkb );
```

```
407             Mtau = drealvector ( 0, 2*Nwkb ); init_realvector ( Mtau, 0, 2*Nwkb );
408             Rlp = dmatrix ( 0, 2*Nwkb, 0, N ); init_realmatrix ( Rlp, 0, 2*Nwkb, 0,
                  N );
409             /* 屈折率分布, 比屈折率分布および波長微分分布 */
410             if (gi == 0) {
411                 for ( j = 0; j <= N; j++ ) { GI[j] = n1*sqrt ( 1.0-2.0*delta*pow
                      (((double) j / (double) N), g) ); }
412                 for ( j = 0; j <= N; j++ ) { q[j] = (GI[j]*GI[j] - n0*n0) / (n1*n1 -
                      n0*n0); }
413                 for ( j = 0; j <= N; j++ ) { q2[j] = GI[j]*GI[j] - (lamda*GI[j]*dndl
                      (lamda*1.0e9, GI[j], mater)) / (1 - (lamda/GI[j])*dndl
                      (lamda*1.0e9, GI[j], mater)); }
414                 if ( y == Nl/2 || fout == 1 ) { for ( j = 0; j <= N; j++ ) { fprintf
                      ( fp3, "%f, %f, %f, %f¥n", A*1.0e6* (double) j / (double) N, GI[j],
                       q[j], q2[j] ); }}}
415             if (gi == 1) {
416                 for ( j = 0; j <= N; j++ ) { GI[j] = dndc ( lamda*1.0e9, mater )*GC
                      [j] + nclad ( lamda*1.0e9, mater ); } // 評価波長における屈折率分布
                      に換算
417                 for ( j = 0; j <= N; j++ ) { q[j] = (GI[j]*GI[j] - GI[N]*GI[N]) / (GI
                      [0]*GI[0] - GI[N]*GI[N]); }
418                 for ( j = 0; j <= N; j++ ) { q2[j] = GI[j]*GI[j] - (lamda*GI[j]*dndl
                      (lamda*1.0e9, GI[j], mater)) / (1 - (lamda/GI[j])*dndl
                      (lamda*1.0e9, GI[j], mater)); }
419                 if ( y == Nl/2 || fout == 1 ) { for ( j = 0; j <= N; j++ ) { fprintf
                      ( fp3, "%f, %f, %f, %f, %f¥n", A*1.0e6* (double) j / (double) N, GC
                      [j], GI[j], q[j], q2[j] ); }}}
420
421             //! 評価条件の出力
422             if (  y == 0 ) {
423                 fprintf ( fp2, "Material, mater, %d¥n", mater );
424                 fprintf ( fp2, "Index exponent, g, %.2lf¥n", gsingle );
425                 fprintf ( fp2, "Central wavelength, λ0, %f nm¥n", lamda0*1e9 );
426                 fprintf ( fp2, "Step size of wavelength, dl, %f nm¥n", dl*1e9 );
427                 fprintf ( fp2, "Partition number of evaluated wavelength, Nl, %d¥n",
                      Nl );
428                 fprintf ( fp2, "Core radius, A, %f um¥n", A*1e6 );
429                 fprintf ( fp2, "Analysis region in radial axis, AA,%f um¥n",
                      AA*1e6 );
430                 fprintf ( fp2, "Refractive index at the core center, n1,%f¥n", n1 );
431                 fprintf ( fp2, "Refractive index in the cladding, n0,%f¥n", n0 );
432                 fprintf ( fp2, "Relative refractive index, Δ,%f¥n", delta );
433                 fprintf ( fp2, "Numerical aperture, NA, %f¥n", NA );
434                 fprintf ( fp2, "Step size of the elements, dr, %f, nm¥n", dr*1e9 );
435                 fprintf ( fp2, "Step size of propagation constant, dβ,%f¥n",
                      dbeta );
436                 fprintf ( fp2, "Partition number of propagation constant, Nβ,%d¥n",
                      Nbeta );
437                 fprintf ( fp2, "Partition number of fiber core radius, N,%d¥n", N );
438                 fprintf ( fp2, "Partition number of fiber cladding, Nclad,%d¥n",
                      Nclad );
439                 fprintf ( fp2, "Maximum allowable error for convergence solution
                      vector, eps1,%f¥n", eps1 );
440                 fprintf ( fp2, "Maximum allowable error of zero eigen value, eps2,%f
                      ¥n", eps2 );
441                 fprintf ( fp2, "Maximum number of iterations in inverse power method,
                       jmax,%d¥n", jmax );
442                 fprintf ( fp2, "Correlation length, Dc, %e, nm¥n", Dc*1e9 );
443                 fprintf ( fp2, "Mean square of dielectric constant fluctuation,
                      <dε2>, %e¥n", deps2 );
444                 fprintf ( fp2, "Total fiber length,zmax,%e,m¥n", zmax );
445                 fprintf ( fp2, "Step size of fiber length,dz,%e,m¥n", dz );
446                 fprintf ( fp2, "Step size of time,Tv,%e,ps¥n", Tv );
447                 fprintf ( fp2, "Total spatial steps,Nz,%d¥n", Nz );
```

```
448                 fprintf ( fp2, "File output step interval,Nzout,%d¥n", Nzout );
449                 fprintf ( fp2, "Minimum evaluated frequency,fminx,%e,GHz¥n",
                      fmin*1.0e3 );
450                 fprintf ( fp2, "Maximum evaluated frequency,fmax,%e,GHz¥n",
                      fmax*1.0e3 );
451                 fprintf ( fp2, "Step size of evaluated frequency,df,%e,GHz¥n",
                      df*1.0e3 );
452                 printf ( "Central wavelength λ0: %f nm¥n", lamda0*1e9 );
453                 printf ( "Step size of wavelength dl: %f nm¥n", dl*1e9 );
454                 printf ( "Partition number of evaluated wavelength Nl: %d¥n", Nl );
455                 printf ( "Minimum evaluated spectral frequency, fpmin, %f THz¥n",
                      fpmin );
456                 printf ( "Maximum evaluated spectral frequency, fpmax, %f THz¥n",
                      fpmax );
457                 printf ( "Core radius A: %f um¥n", A*1.0e6 );
458                 printf ( "Analysis region AA: %f um¥n", AA*1e6 );
459                 printf ( " Refractive index at the core center n1: %f¥n", n1 );
460                 printf ( "Refractive index in the cladding n0: %f¥n", n0 );
461                 printf ( "Relative refractive index Δ: %f¥n", delta );
462                 printf ( "Numerical aperture NA: %f¥n", NA );
463                 printf ( "Step size of the elements dr: %f nm¥n", dr*1.0e9 );
464                 printf ( "Partition number of fiber core radius N: %d¥n", N );
465                 printf ( "Partition number of fiber cladding Nclad: %d¥n", Nclad );
466                 printf ( "Step size of propagation constants dβ: %f¥n", dbeta );
467                 printf ( "Partition number of propagation constants Nβ: %d¥n",
                      Nbeta );
468                 printf ( "Maximum allowable error for convergence solution vector
                      eps1: %f¥n", eps1 );
469                 printf ( "Maximum allowable error of zero eigen value eps2: %f¥n",
                      eps2 );
470                 printf ( "Maximum number of iterations in inverse power method jmax:
                      %d¥n", jmax );
471                 printf ( "Correlation length Dc: %e m¥n", Dc );
472                 printf ( "Mean square of dielectric constant fluctuation <dε2>: %f
                      ¥n", deps2 );
473                 printf ( "Total fiber length zmax: %e m¥n", zmax );
474                 printf ( "Step size of fiber length dz: %e m¥n", dz );
475                 printf ( "Step size of time Tv: %e ps¥n", Tv );
476                 printf ( "Total spatial steps Nz: %d¥n", Nz );
477                 printf ( "File output step interval Nzout: %d¥n", Nzout );
478                 printf ( "Minimum evaluated frequency fminx: %e GHz¥n", fmin*1.0e3 );
479                 printf ( "Maximum evaluated frequency fmax: %e GHz¥n", fmax*1.0e3 );
480                 printf ( "Step size of evaluated frequency df: %e GHz¥n¥n",
                      df*1.0e3 ); }
481               printf ( "Wavelength: %f nm (%d/%d)¥n", lamda*1.0e9,y,Nl );
482               if ( wo ==0 ) { printf ( "without mode coupling¥n"); }
483               if ( wo ==1 ) { printf ( "with microscopic heterogeneities¥n"); }
484               if ( wo ==2 ) { printf ( "with microbending¥n"); }
485               fprintf ( fp2, "lamda=%f nm¥n", lamda*1.0e9 );
486               fprintf ( fp4, "lamda=%f nm¥n", lamda*1.0e9 );
487               if (  y == Nl/2 || fout == 1 ) {
488                   fprintf ( fp4, "m,l,p,tau,beta,ne,confinement,R_infinite,eig,");
489                   for ( j = 0; j <= N; j++ )  { fprintf ( fp4,"%f,", (double)
                        j*dr*1.0e6 ); } fprintf ( fp4,"¥n" ); }
490               fprintf ( fq, "lamda=%f nm¥n", lamda*1.0e9 );
491               fprintf ( fr2, "lamda=%f nm¥n", lamda*1.0e9 );
492               fprintf ( fr3, "lamda=%f nm¥n", lamda*1.0e9 );
493
494
495
    //! ////////////////////////////////////////////////////////////
496             //! //////////          2. モード解析                  //////////
                  (FEM.cpp)
497
    //! ////////////////////////////////////////////////////////////
```

```
498             //
499             for ( m = 0 ; ; m++ ) {
500                 l = 1; beta = k*n1;
501                 /*****************************************************************/
502                 for ( ; ; )  {
503                     /* 係数行列計算および改訂コレスキー分解 */
504                     w = aa*sqrt ( (beta*beta) - (k*k)*(n0*n0) );
505                     S_matrix ( a, b, q, m, N, v, w, D );
506                     mcholesky ( a, b, ML, MD, m, N );
507                     /* 初期ベクトル R0 の付与 */
508                     R0 ( MD, R, m, N );
509                     /*  連立一次方程式 SR=(LDL)R=bR の反復評価 */
510                     for ( j = 0 ; ; j++ ) {
511                         for ( i = 0; i <= N; i++ )  { Rb[i] = R[i]; }
512                         mcholesky_sol ( ML, MD, R, m, N );
513                         R_norm ( R, N );
514                         /* 収束判定 */
515                         dd = 0, ds = 0;
516                         for ( i = 0; i <= N; i++ ) {
517                             dd = dd + (Rb[i] - R[i])*(Rb[i] - R[i]);
518                             ds = ds + (Rb[i] + R[i])*(Rb[i] + R[i]); }
519                         if ( dd < eps1 || ds < eps1 ) break;
520                         if ( j >= jmax ) goto next;
521                         // ① RとRbの成分差ddが0に漸近すれば収束 (break) .
522                         // ② RとRbの成分和dsが0に漸近すれば中止 (break) .
523                         // ③ 反復回数が上限値 jmax を超えたらβを変更して再計算 (go to next) .
524                     }
525                     /* 固有値の計算 */
526                     eig = Eigen ( R, a, b, N, m );
527                     /* 固有値の妥当性評価 */
528                     // 「収束固有値 eig が前回値 bb と同値」であればβを変えて初めから再計算
529                     if ( eig == bb ) {
530                         dbeta = k*(n1 - n0) / (double) Nbeta;
531                         beta = beta - 1.0*dbeta;
532                         continue; }
533
534                     //  ①「0< 収束固有値 eig < eps2」であれば零固有値として採用
535                     if ( 0.0 < eig && eig < eps2 )  {
536                         /* 横方向電場成分Rの規格化  (パワーを1Wとする) */
537                         sum = sumcore = sumclad = 0.0;
538                         for ( j = 0; j < N; j++ ) { sumcore = sumcore + R[j]*R[j]*(j*dr)*dr; }
539                         for ( j = 0; j < Nclad; j++ ) { sumclad = sumclad + R[N]*( bessk (m, w*(j*dr+A)) / bessk (m, w*A) )*R[N]*( bessk (m, w*(j*dr+A)) / bessk (m, w*A) )*(j*dr+A)*dr; }
540                         for ( j = 0; j <= N; j++ ) { R2[j] = R[j] * sqrt ( (omega*Mu0) / (PI*beta*(sumcore + sumclad)) ); }
541                         tau = (1.0 / (C*1.0e-12))*(k / beta) * dbdk_bunshi ( R, q2, D, w, m, N ) / dbdk_bunbo ( R, D, w, m, N ); // = (1/c)*(dβ/dk) [ps/m]
542                         /* 計算結果の出力 */
543                         Rinf = R[N]*( bessk (m, w*(Nclad*dr+A)) / bessk (m, w*A) );
544                         if ( y == Nl/2 || fout == 1 ) { fprintf ( fp4, "%d, %d, %d, %f, %f, %f, %f, %f,",m,l,2*l+m-1,tau,beta,beta/k,sumcore/(sumcore+sumclad),Rinf,eig ); }
545                         modem[NLP] = m, model[NLP] = l, Mtau[NLP] = tau, Mbeta[NLP] = beta;
546                         modep[NLP] = 2*l + m - 1; if ( Ptotal < modep[NLP] ) { Ptotal = modep[NLP]; } // 主モード次数
547                         for ( j = 0; j <= N; j++ )  { if ( fout == 1 || y == Nl/2 ) { fprintf ( fp4,"%f,", R2[j] ); } Rlp[NLP][j] = R2[j]; }
548                         if ( y == Nl/2 || fout == 1 ) { fprintf ( fp4,"¥n" ); }
```

```
549                         //printf ( "%d, %d, %f, %f, %f, %f, %f¥n", m, l, tau, beta,
                          beta/k, eig, R[N] );
550
551                         dbeta = k*(n1 - n0) / (double) Nbeta;
552                         bb = eig;
553                         l = l + 1;
554                         NLP = NLP + 1; if ( m == 0 ) { NLP0 = NLP0 +1;}
555                         count = 0; }
556
557                     //  ②「0 < 収束固有値 eig」かつ「-1 < 前回値 bb < 0」であれば
558                     if ( eig > 0.0 && bb < 0.0 && (fabs(bb) < 1.0) ) {
559                         beta = beta + dbeta;
560                         dbeta = dbeta / 2.0;
561                         count = count + 1; }
562
563                     // ③ その他
564                     else { bb = eig; count = 0; }
565
566                     if ( count > 1000 ) {
567                         dbeta = k*(n1 - n0) / (double) Nbeta;
568                         beta = beta - dbeta; }
569 next:
570                     beta = beta - dbeta;
571                     if ( beta < k*n0 )  break;
572                 }
573                 /*****************************************************************/
574                 if ( l == 1 ) { Ntotal = ( NLP0 )*2 + ( NLP - NLP0 )*4; break; } //
                    mが最高次数に到達
575             }
576
577             free_drealvector ( GI, 0 ); free_drealvector ( q, 0 ); free_drealvector
                ( q2, 0 );
578             free_drealvector ( R, 0 ); free_drealvector ( R2, 0 ); free_drealvector
                ( Rb, 0 );
579             free_drealvector ( a, 0 ); free_drealvector ( b, 0 );
580             free_drealvector ( ML, 0 ); free_drealvector ( MD, 0 );
581
582             printf ( "Total numbers of LPml modes (WKB) Nwkb: %d¥n", Nwkb);
583             printf ( "Total numbers of LPml modes NLP: %d¥n", NLP);
584             printf ( "Total numbers of LP0l modes NLP0: %d¥n", NLP0);
585             printf ( "Total numbers of all modes Ntotal: %d¥n", Ntotal);
586             printf ( "Total numbers of all mode groups Ptotal: %d¥n", Ptotal);
587
588
//! ////////////////////////////////////////////////////////////
589             //! //////////          3. LPモード特性の整
                理                ///////////
590
//! ////////////////////////////////////////////////////////////
591             /* 群遅延範囲 */
592             // 各ループの単位長群遅延範囲 (taumin < tau < taumax)
593             taumax = taumin = Mtau[0];
594             for ( myu = 0; myu < NLP; myu++ ) {
595                 if ( taumax < Mtau[myu] ) { taumax = Mtau[myu]; }
596                 if ( taumin > Mtau[myu] ) { taumin = Mtau[myu]; }}
597             // 各ループの最高次モード群単位長群遅延
598             nctaumax = taumin;
599             for ( myu = 0; myu < NLP; myu++ ) {
600                 if ( modep [myu] == Ptotal ) {  if ( nctaumax < Mtau[myu] )
                      { nctaumax = Mtau[myu]; }}}
601             // 各ループの基準時間補正要素数 (nstd)
602             if ( y == 0 ) { tauminstd = taumin; }
603             nstd = (int) ( (taumin - tauminstd)*zmax / Tv );
604             printf ("nstd:%d,nstdmin:%d¥n", nstd,nstdmin);
605             fprintf ( fp2, "Minimum group delay per unit length,taumin,%e,ps/m¥n",
                taumin );
```

```
606             fprintf ( fp2, "Maximum group delay per unit length,taumax,%e,ps/m¥n",
                  taumax );
607             // 各ループの最大郡遅延差 (Lmax)
608             Lmax = (int) ( (( taumax - taumin )*( (double)Nz*dz ) / Tv) + 0.5 );
609             fprintf ( fp2, "Total time steps,Lmax,%d¥n", Lmax );
610             if ( Lmax > ( 1.0e9 / (8.0*NLP) ) ) { printf ( "Memory over!¥n" ); exit
                  ( EXIT_FAILURE ); }
611             printf ( "Required memory for CPE analysis: %fGB¥n", (double)
                  (Lmax*NLP*8) /1.0e9 );
612             // プロファイルループの最大郡遅延差 (nmax)
613             if ( Lmax > nmax ) { nmax = Lmax; }
614             fprintf ( fp2, "Maximum time step,nmax,%d¥n", nmax );
615             // インパルス応答格納配列数 (Pnum)
616             if ( nstd == 0 ) { Pnum [y] = nmax + ( nstdmax - nstdmin ); }
617             if ( nstd > 0 ) {
618                 if ( nstd < nstdmax ) { Pnum [y] = nmax + ( nstdmax - nstdmin ); }
619                 else { Pnum [y] = nmax + ( nstdmax - nstdmin ) + ( nstd -
                      nstdmax ); }}
620             if ( nstd < 0 ) {
621                 if ( nstd < nstdmin ) { Pnum [y] = nmax + ( nstdmax - nstdmin ) +
                      ( nstdmin - nstd ); }
622                 else { Pnum [y] = nmax + ( nstdmax - nstdmin ); }}
623             fprintf ( fp2, "Net total time steps,Pnum,%d¥n", Pnum [y] );
624
625             /* 配列の記憶領域確保および初期化 */
626             H = dmatrix ( 0, NLP-1, 0, NLP-1 );         init_realmatrix ( H, 0,
                  NLP-1, 0, NLP-1 );
627             Amin = dmatrix ( 0, NLP-1, 0, Lmax );       init_realmatrix ( Amin, 0,
                  NLP-1, 0, Lmax );
628             Amplu = dmatrix ( 0, NLP-1, 0, Lmax );      init_realmatrix ( Amplu, 0,
                  NLP-1, 0, Lmax );
629             alpha = drealvector ( 0, NLP-1 );           init_realvector ( alpha, 0,
                  NLP-1 );
630             kim = dintvector ( 0, NLP-1 );              init_intvector ( kim, 0,
                  NLP-1 );
631             Pm = drealvector ( 0, NLP-1 );              init_realvector ( Pm, 0,
                  NLP-1 );
632             Pg = drealvector ( 0, Ptotal-1 );           init_realvector ( Pg, 0,
                  Ptotal-1 );
633
634             if ( matdis == 0 ) { P = drealvector ( 0, Lmax ); init_realvector ( P, 0,
                   Lmax ); }
635             A00 = 0.0;
636
637
    //! ////////////////////////////////////////////////////////////////////////////
    /
638             //! ////////// 4. LPモード μ とLPモード ν 間の電力結合係数計
                  算 //////////
639
    //! ////////////////////////////////////////////////////////////////////////////
640             //!
641             //! H行列の算出
642             if ( wo == 0 ) { for ( myu = 0; myu < NLP; myu++ ) { H[myu][myu] =
                  1.0; } }
643             else {
644             // ミクロ不均一構造
645             if ( wo == 1) { dbmn = 0.0; E_over = 0.0; hmn =0.0;
646 #pragma omp parallel
647             {
648 #pragma omp for
649             for ( myu = 0; myu < NLP; myu++ ) {
650                 for ( nyu = 0; nyu < NLP; nyu++ ) {
651                     dbmn = Mbeta [myu] - Mbeta [nyu];
652                     for ( n = 1; n <= N; n++ ) {
```

```
653                         E_over = E_over + (2.0*PI)*( Rlp[myu][n]*Rlp[nyu][n] )*( Rlp[myu][n]*Rlp[nyu][n] )*((double)(n-1)*dr)*dr; }
654
655                     if ( modep[myu] == Ptotal || modep[nyu] == Ptotal ) { hmn = 0.0; }  // 最高次モードは無視
656                     else {
657                         hmn = deps2*((omega*omega*(PI*sqrt(PI))*(Dc*Dc*Dc)) / 8.0) *exp(-(dbmn*dbmn)*(Dc*Dc) / 4.0)*E_over; }
658
659                     if ( myu == nyu ) { H[myu][nyu] = 0.0; } else { H[myu][nyu] = hmn*dz; } // 仮入力
660
661                     if ( y == Nl/2 || fout == 1 ) {
662                         if ( myu != nyu && myu > nyu ) {
663 //                      if ( myu != nyu ) {
664                             fprintf ( fq, "%d,%d,%d,%d,%f,%d,%d,%d,%d,%f,%d,%f,%e,%e¥n",    myu, modem [myu], model [myu], modep [myu], Mbeta [myu],
665                                 nyu, modem [nyu], model [nyu], modep [nyu], Mbeta [nyu], abs(modep[myu]-modep[nyu]), fabs(dbmn), E_over, hmn ); }}
666                     E_over = 0.0; }}
667             }
668             }
669             // マイクロベンディング
670             if ( wo == 2) { dbmn = 0.0; hmn =0.0;
671             for ( myu = 0; myu < NLP; myu++ ) {
672                 for ( nyu = 0; nyu < NLP; nyu++ ) {
673                     dbmn = Mbeta [myu] - Mbeta [nyu];
674
675                     if ( modep[myu] == Ptotal || modep[nyu] == Ptotal ) { hmn = 0.0; }  // 最高次モードは無視
676                     else if ( abs (modep [myu] - modep [nyu]) == 1 ) {
677                         if ( modep[myu] > modep [nyu] ) { mm = modep[myu] ; } else { mm = modep[nyu]; }
678                         hmn = (1.0/8.0)*(n1*k*A)*(n1*k*A)*pow( ((double)mm/(double)Ptotal), 4.0/(g+2.0) )
679                                 *sigma2*sqrt(PI)*Db*exp(-(dbmn*dbmn)*(Db*Db) / 4.0); }
680                     else { hmn = 0.0; }
681
682                     if ( myu == nyu ) { H[myu][nyu] = 0.0; } else { H[myu][nyu] = hmn*dz; } // 仮入力
683                     if ( y == Nl/2 || fout == 1 ) {
684                         if ( myu != nyu && myu > nyu ) {
685 //                      if ( myu != nyu ) {
686                             fprintf ( fq, "%d,%d,%d,%d,%f,%d,%d,  %d,%d,%f,%d,%f,%e¥n", myu, modem [myu], model [myu], modep [myu], Mbeta [myu],
687                                 nyu, modem [nyu], model [nyu], modep [nyu], Mbeta [nyu], abs(modep[myu]-modep[nyu]), fabs(dbmn), hmn ); }}
688                      hmn = 0.0; }}
689             }   //! H行列の算出終了
690
691             // 初期化
692             hmn = 0.0; myu = 0; nyu = 0;
693             // 対格要素
694             for ( myu = 0; myu < NLP; myu++ ) { Hrowsum = 0.0;
695             for ( nyu = 0; nyu < NLP; nyu++ ) {
696                 if ( modem[nyu] == 0 ) { Hrowsum = Hrowsum + H[myu][nyu]; } else { Hrowsum = Hrowsum + 2.0*H[myu][nyu]; }}
697             H[myu][myu] = 1.0 - ( 2.0*alpha[myu]*dz + Hrowsum ); }
698             // 非対格要素
699             for ( myu = 0; myu < NLP; myu++ ) {
700             for ( nyu = 0; nyu < NLP; nyu++ ) {
701                 if ( myu != nyu ) { if ( modem[myu] == 0 ) { H[myu][nyu] = 1.0*H[myu][nyu]; } else { H[myu][nyu] = 2.0*H[myu][nyu]; }}}}
702             // 安定条件の確認
703             Hmmmin = H[0][0];
```

```
704             for ( m = 1; m < NLP; m++ ) { if ( Hmmmin > H[m][m] ) { Hmmmin = H[m]
                  [m]; } }
705             if ( Hmmmin < 0) { printf ( "Too large Δz value! Change the value
                  appropriately!¥n" ); exit ( EXIT_FAILURE ); }}
706             /* H行列の出力*/
707             if ( y == Nl/2 && fout == 1 ) {         // ここを編集するべきかもしれせ
                  ん
708                 for ( myu = 0; myu < NLP; myu++ ) {
709                 for ( nyu = 0; nyu < NLP; nyu++ ) { fprintf ( fr, "%f,", H[myu]
                      [nyu] ); } fprintf ( fr, "¥n" ); }}
710
711             //! 励振条件設定（A+行列の算出）
712             // OFL condition
713             if ( launch == 0 ) {
714                 for ( m = 0; m < NLP; m++ ) {
715                     if ( matdis == 0 ) { if (modem[m] == 0) { Amplu[m][0] = 100.0; }
                       else { Amplu[m][0] = 200.0; }} // 縮退数の考慮
716                     if ( matdis == 1 ) {
717                         if (modem[m] == 0) { Amplu[m][0] = 100.0*OSin[ (int)((lamda -
                       lpmin)/dlp) ]; }
718                       else { Amplu[m][0] = 200.0*OSin[ (int)((lamda -lpmin)/
                       dlp) ]; }}}}
719
720             // RML condition
721             if ( launch == 1 ) {
722 #pragma omp parallel
723                 {
724 #pragma omp for
725                 for ( m = 0; m < NLP; m++ ) { Amev = Amod = 0.0;
726                     for ( i = 0; i < Nxy; i++ ) { xx = ( r0 - ( (double)Nxy*dx /
                       2.0 ) ) + (double)i*dx;
727                       for ( j = 0; j < Nxy; j++ ) { yy = - ( (double)Nxy*dy / 2.0 )
                        + (double)j*dy;
728                       rr = sqrt ( xx*xx + yy*yy ); nr = (int) (rr / dr)   ; // 切り
                       捨て？
729                       if ( nr == 0 ) { Rxy =  Rlp [m][0] + ( Rlp [m][1] - Rlp [m]
                       [0]) * (rr /dr); ;}
730 //                      else { Rxy = Rlp [m][nr] + ( Rlp [m][nr+1] - Rlp [m][nr]) *
    ((rr - (double)nr*dr) /dr); } // interpolation
731                       else if ( nr <= N ) { Rxy = Rlp [m][nr] + ( Rlp [m][nr+1] -
                      Rlp [m][nr]) * ((rr - (double)nr*dr) /dr); } // core
                      (interpolation)
732                       else if ( nr > N ) { Rxy = Rlp [m][N]*( bessk (m, w*rr) /
                      bessk (m, w*A) ); } // cladding
733                       Emev = Rxy*cos( (double)(modem[m])*atan2(yy, xx) );
734                       Emod = Rxy*sin( (double)(modem[m])*atan2(yy, xx) );
735                       Ein = exp ( - ((xx - r0)*(xx - r0) + yy*yy) / (w0*w0) ); //
                      input field
736                       Amev = Amev + Emev*Ein*dx*dy; Amod = Amod +
                      Emod*Ein*dx*dy; }}
737                       Amev = Amev*Amev / (((2.0*omega*Mu0)/Mbeta[m])*
                      (PI*w0*w0/2.0));
738                       Amod =  Amod*Amod / (((2.0*omega*Mu0)/Mbeta[m])*
                      (PI*w0*w0/2.0));
739                     if ( matdis == 0 ) { Amplu[m][0] = 100.0* (Amod + Amev); }
740                     if ( matdis == 1 ) { Amplu[m][0] = 100.0*( Amod + Amev )*OSin
                       [ (int)((lamda -lpmin)/dlp) ]; }
741                     //printf ("Amplu [%d][0] =%f¥n", modem[m],Amplu [m][0] );
742                     }}}
743
744             //TODO For measuring
                  minEMBc   ////////////////////////////////////////
745             if ( launch == 2 ) {
746                 //! 貼り付けただけ部分はここか
                    ら   ////////////////////////////////////////
```

```
747             /*if ((fp7 = fopen("FEM_LPmode_VCSELinput.csv", "r")) != NULL) {
                  //      VCSEL のLPモードの1次元強度分布ファイルを開く
748
749             int minput;
750             char MPDfilename[256];
751             char coupletype[128];
752             if (couple == 0) { sprintf(coupletype, "V2F"); }
753             else if (couple == 1) { sprintf(coupletype, "F2F"); }
754             else { sprintf(coupletype, "error_warning"); }
755             if (couple == 0) { max = num_mode_vin; }
756             else if (couple == 1) { max = NLP; }
757             for (int ii = 0; ii <= OFFrange / OFFres + 1; ii++) {
758                 if (launch == 2) { printf("¥ncalculating MPD for lateral offset...
                      ¥n"); }
759                 if (launch == 3) { printf("¥ncalculating MPD for lateral offset of %d
                      um...¥n¥n", ii * OFFres); }
760                 fscanf(fp7, "%s", trash);
761                 double E2m_evin, E2m_ev;
762                 for (minput = 0; minput < max; minput++) {
763                     if (ii == 0) {
764                         fscanf(fp7, "%d, %d, %lf, %lf, %lf, %lf, %lf,", &min, &lin,
                          &tauin, &betain, &betain_devided_by_k, &Rinfin, &eigin);
765                         modemin[minput] = min;
766                         modelin[minput] = lin;
767                         Mbetain[minput] = betain;           }
768                     printf("%d¥t%d¥n", modemin[minput], modelin[minput]);
769                     if (couple == 0) { Adash = Avin; }      if (couple == 1) { Adash
                      = A; }
770
771                     for (j = 0; j <= (int)(Adash / dr); j++) {
772                         if (ii == 0) {
773                             fscanf(fp7, "%lf,", &Rinlp[minput][j]);     }}
774                     if (ii == 0) { fscanf(fp7, "%lf¥n", &Rinlp[minput][j]); }
775                     for (m = 0; m < NLP; m++) {
776                         Am_evev = 0.0;      Am_evod = 0.0;
777                         Am_odev = 0.0;      Am_odod = 0.0;
778                         E2m_evin = 0.0;     E2m_ev = 0.0;
779                         Rinxy = 0.0;
780
781                         for (i = 0; i < Nxy; i++) {
782                             xx1 = (-(double)Nxy*dx / 2.0) + ((double)i*dx);
783                             if (launch == 2) {
784                                 xx2 = (r0 - (double)Nxy*dx / 2.0) + ((double)i*dx);
                                  }
785                             if (launch == 3) {
786                                 xx2 = ((ii * (double)OFFres * 1.0e-6) - (double)
                              Nxy*dx / 2.0) + ((double)i*dx); }
787
788                             for (j = 0; j < Nxy; j++) {
789                                 yy = (-(double)Nxy*dx / 2.0) + ((double)j*dy);
790                                 rr1 = sqrt(xx1*xx1 + yy * yy);
791                                 rr2 = sqrt(xx2*xx2 + yy * yy);
792                                 nrr1 = (int)(rr1 / dr);
793                                 nrr2 = (int)(rr2 / dr);
794
795                                 //   ここは入射するレーザ（ファイバ）の，xy平面上の2
                              次元強度分布を計算
796                                 if (nrr1 == 0) {        // 以下の配列Rinxyは，メモリ
                              内の配列Rlpを使うので読み込みはHDDを介すより速い
797                                     Rinxy = Rinlp[minput][0] + (Rinlp[minput][1] -
                              Rinlp[minput][0]) * (rr1 / dr);
798                                 }
799                                 else if (nrr1 <= Nvin) {
800                                     Rinxy = Rinlp[minput][nrr1] + (Rinlp[minput][nrr1
                               + 1] - Rinlp[minput][nrr1]) * ((rr1 - (double)nrr1*dr) / dr);
801                                 }
```

```
802                             else if (nrr1 > Nvin) {
803                                 //printf("%d, %lf, %d, %lf¥n", minput, Mbetain
    [minput], Nvin, Avin);
804                                 win = aa * sqrt(Mbetain[minput] * Mbetain[minput]
     - k * k*n0*n0);
805                                 Rinxy = RinIp[minput][Nvin] * (bessk(modemin
    [minput], win*rr1) / bessk(modemin[minput], win*Avin));
806                             }
807                             //   ここは受け手側のファイバにおける，xy平面上の2次
    元強度分布を計算                    /// 以下の配列Rinxyは，メモリ内
    の配列RIpを使うので読み込みはHDDを介すより速い
808                             if (nrr2 == 0) {
809                                 Rxy = RIp[m][0] + (RIp[m][1] - RIp[m][0]) *
    (rr2 / dr);
810                             }
811                             else if (nrr2 <= N) {
812                                 Rxy = RIp[m][nrr2] + (RIp[m][nrr2 + 1] - RIp[m]
    [nrr2]) * ((rr2 - (double)nrr2*dr) / dr);   }
813                             else if (nrr2 > N) {
814                                 w = aa * sqrt(Mbeta[m] * Mbeta[m] - k * k*n0*n0);
815                                 Rxy = RIp[m][N] * (bessk(modem[m], w*rr2) / bessk
    (modem[m], w*A));                               }
816
817                             //   重なり積分に用いる電場分布の算出・重なり積分の実
    行
818                             Em_evin = Rinxy * cos((double)(modemin[minput]) *
    atan2(yy, xx1));
819                             Em_odin = Rinxy * sin((double)(modemin[minput]) *
    atan2(yy, xx1));
820                             Em_ev = Rxy * cos((double)(modem[m]) * atan2(yy,
    xx2));
821                             Em_od = Rxy * sin((double)(modem[m]) * atan2(yy,
    xx2));
822                             if (i < 1 && j < 1) {

823                                 if (modem[m] == 0) {
824                                     //printf("%lf, %lf, %lf, %lf¥t¥t", Em_evin,
    Em_odin, Em_ev, Em_od);
825                                     printf("%d ", m);       }
826                                 else { printf("%d ", m); }
827                             }
828                             E2m_evin = E2m_evin + Em_evin * Em_evin*dx*dy;
829                             E2m_ev = E2m_ev + Em_ev * Em_ev*dx*dy;
830                             Am_evev = Am_evev + Em_evin * Em_ev*dx*dy;
831                             Am_evod = Am_evod + Em_evin * Em_od*dx*dy;
832                             Am_odev = Am_odev + Em_odin * Em_ev*dx*dy;
833                             Am_odod = Am_odod + Em_odin * Em_od*dx*dy;          }
834                         }
835                         cef_evev = Am_evev * Am_evev / (E2m_evin * E2m_ev);
836                         cef_evod = Am_evod * Am_evod / (E2m_evin * E2m_ev);
837                         cef_odev = Am_odev * Am_odev / (E2m_evin * E2m_ev);
838                         cef_odod = Am_odod * Am_odod / (E2m_evin * E2m_ev);
839
840                         if (modem[m] == 0 && modemin[minput] == 0) {
841                             MPD2d[m][minput] = 100.0*(cef_evev + cef_evod + cef_odev
    + cef_odod);                                }
842                         if (modem[m] != 0 && modemin[minput] == 0) {
843                             MPD2d[m][minput] = 100.0*(cef_evev + cef_evod + cef_odev
    + cef_odod) / 2;                            }
844                         if (modem[m] == 0 && modemin[minput] != 0) {
845                             MPD2d[m][minput] = 100.0*(cef_evev + cef_evod + cef_odev
    + cef_odod) / 2;                            }
846                         if (modem[m] != 0 && modemin[minput] != 0) {
847                             MPD2d[m][minput] = 100.0*(cef_evev + cef_evod + cef_odev
    + cef_odod) / 2;                            }
848                     }
```

```
849                     printf("¥n");
850                 }
851                 printf("¥n");
852                 for (minput = 0; minput < max; minput++) { printf("%d¥t%lf¥n",
                      modemin[minput], Mbetain[minput]); }
853                 printf("¥n");
854                 for (m = 0; m < NLP; m++) { printf("%d¥t%lf¥n", modem[m], Mbeta
                      [m]); }
855                 if (launch == 2) { sprintf(MPDfilename, "MPD_%s¥¥MPD2d_%s_%
                      dum_Offset_single(not_dependence_on_x).csv", coupletype,
                      coupletype, r0); }
856                 if (launch == 3) { sprintf(MPDfilename, "MPD_%s¥¥MPD2d_%s_%
                      dum_Offset.csv", coupletype, coupletype, ii*OFFres); }
857                 if ((fp8 = fopen(MPDfilename, "w")) != NULL) {
858                     fprintf(fp8, "Mode number for receiver-side fiber,");
859                     if (couple == 0) { fprintf(fp8, "m, l, "); }
860                     for (j = 0; j < max; j++) { fprintf(fp8, "%d,", j); }
861                     fprintf(fp8, "¥n");
862                     for (i = 0; i < NLP; i++) {
863                         fprintf(fp8, "%d,", i);
864                         if(couple == 0) { fprintf(fp8, "%d, %d, ", modem[i], model
                          [i]); }
865                         for (j = 0; j < max; j++) {
866                             // printf("%e, ", MPD2d[i][j]);
867                             fprintf(fp8, "%e,", MPD2d[i][j]);       }
868                         fprintf(fp8, "¥n");             }
869                     fclose(fp8);                }
870                 if (launch == 2) break;     }
871           }*/     //! ここまで
                    で  ////////////////////////////////////////////////
872                 for (n = 0;n < Nom; n++) {
873                 {
874                     for ( m = 0; m < NLP; m++ ) { Amev = Amod = 0.0;
875                         for ( i = 0; i < Nxy; i++ ) { xx = ( r0 - ( (double)Nxy*dx /
                          2.0 ) ) + (double)i*dx;
876                             for ( j = 0; j < Nxy; j++ ) { yy = - ( (double)Nxy*dy /
                          2.0 ) + (double)j*dy;
877                             rr = sqrt ( xx*xx + yy*yy ); nr = (int) (rr / dr)   ; //
                          切り捨て？
878                             if ( nr == 0 ) { Rxy =  Rlp [m][0] + ( Rlp [m][1] - Rlp
                          [m][0]) * (rr /dr); ;}
879     //                          else { Rxy = Rlp [m][nr] + ( Rlp [m][nr+1] - Rlp [m][nr])
            * ((rr - (double)nr*dr) /dr); } // interpolation
880                             else if ( nr <= N ) { Rxy = Rlp [m][nr] + ( Rlp [m][nr+1]
                           - Rlp [m][nr]) * ((rr - (double)nr*dr) /dr); } // core
                          (interpolation)
881                             else if ( nr > N ) { Rxy = Rlp [m][N]*( bessk (m, w*rr) /
                           bessk (m, w*A) ); } // cladding
882                             Emev = Rxy*cos( (double)(modem[m])*atan2(yy, xx) );
883                             Emod = Rxy*sin( (double)(modem[m])*atan2(yy, xx) );
884                             Ein = exp ( - ((xx - r0)*(xx - r0) + yy*yy) /
                          (w0*w0) ); // input field
885                             Amev = Amev + Emev*Ein*dx*dy; Amod = Amod +
                          Emod*Ein*dx*dy; }}
886                             Amev = Amev*Amev / (((2.0*omega*Mu0)/Mbeta[m])*
                          (PI*w0*w0/2.0));
887                             Amod =  Amod*Amod / (((2.0*omega*Mu0)/Mbeta[m])*
                          (PI*w0*w0/2.0));
888                         if ( matdis == 0 ) { Amplu[m][0] = 100.0* (Amod + Amev); }
889                         if ( matdis == 1 ) { Amplu[m][0] = 100.0*( Amod + Amev )*0Sin
                          [ (int)((lamda -lpmin)/dlp) ]; }
890                         //printf ("Amplu [%d][0] =%f¥n", modem[m],Amplu [m][0] );
891                         }}}}
892             //TODO End of lanching condition setting for measuring
                  minEMBc   ////////////////////////////////////////////
```

```
893
894             // Correction for the highest mode group ( elimination of power )
895             for ( m = 0; m < NLP; m++ ) { if ( modep[m] == Ptotal ) { Amplu[m][0] =
                  0.0; }}
896             // Total power of input impulse
897             for ( m = 0; m < NLP; m++ ) { A00 = A00 + Amplu[m][0]; } Aw00 = Aw00 +
                  A00;
898
899             fprintf ( fp2, "Amplitude of input impulse,A00,%e,¥n", A00 );
900             fprintf ( fp2, "Cumulative amplitude of input impulse,Aw00,%e,¥n¥n",
                  Aw00 );
901             free_dmatrix ( Rlp, 0, 2*Nwkb, 0, N-1 );
902
903             /* 出力ファイル */
904             if ( y == Nl/2 || fout == 1 ) {
905                 // 時間波形 P
906                 fprintf ( fr2, "nstc=%d,", nstd );
907                 for ( n = 0; n <= Lmax; n++ ) { fprintf ( fr2, "%f,", (double)
                      n*Tv );} fprintf ( fr2, "pct.¥n" );
908                 fprintf ( fr2, "%f,%f¥n", 0.0, A00 );
909                 // モードパワー分布 Pm
910                 fprintf ( fr3, "myu," ); for ( m = 0; m < NLP; m++ ) { fprintf ( fr3,
                      "%d,", m ); } fprintf ( fr3, "¥n" );
911                 fprintf ( fr3, "LP," ); for ( m = 0; m < NLP; m++ ) { fprintf ( fr3,
                      "LP%d_%d,", modem[m], model[m] ); } fprintf ( fr3, "¥n" );
912                 fprintf ( fr3, "m," ); for ( m = 0; m < NLP; m++ ) { fprintf ( fr3,
                      "%d,", modem[m] ); } fprintf ( fr3, "¥n" );
913                 fprintf ( fr3, "l," ); for ( m = 0; m < NLP; m++ ) { fprintf ( fr3,
                      "%d,", model[m] ); } fprintf ( fr3, "¥n" );
914                 fprintf ( fr3, "p," ); for ( m = 0; m < NLP; m++ ) { fprintf ( fr3,
                      "%d,", modep[m] ); } fprintf ( fr3, "¥n" );
915                 fprintf ( fr3, "%f,", 0.0 );
916                 for ( m = 0; m < NLP; m++ ) { for ( n = 0; n <= Lmax; n++ ) { Pm[m] =
                      Pm[m] + Amplu[m][n]; }
917                 fprintf ( fr3, "%f,", Pm[m] ); } fprintf ( fr3,"¥n");
918                 // モード群パワー分布 Pg
919                 fprintf ( fr9, "," ); for ( j = 1; j <= Ptotal; j++ ) { fprintf
                      ( fr9, "%d,", j ); } fprintf ( fr9, "¥n" ); // モード群番号

920                 fprintf ( fr9, "," ); for ( i = 1; i <= Ptotal; i++ ) { count =0;
921                 for ( myu = 0; myu < NLP; myu++ ) { if ( modep [myu] == i ) { count =
                      count +1; }} // 縮退数（モード数）
922                 fprintf ( fr9, "%d,", count ); pdeg[i-1] =count; } fprintf
                      ( fr9,"¥n");
923                 fprintf ( fr9, "%f,", 0.0 );
924                 for ( m = 0; m < NLP; m++ ) { Pg[modep[m]-1] = Pg[modep[m]-1] + Pm
                      [m]; } // モード群パワー
925                 for ( j = 0; j < Ptotal; j++ ) { fprintf ( fr9, "%f,", Pg[j] /
                      (double)pdeg[j] ); Pg[j] = 0.0; }  fprintf ( fr9,"¥n");
926             }
927
928             ////! /////        CONTINUE文        //////
929             ////printf("%lf¥t", OSin[(int)((lamda - lpmin) / dlp)]);
930             ////    if (OSin[(int)((lamda - lpmin) / dlp)] == 0.0) {
931             ////    //contct += 1;       printf("continue count is %d¥n", contct);
932             ////    continue;   }
933
934             //! ////////////////////////////////////////////////////
935             //! //////////          5. 光波伝搬解析          ///////////
936             //! ////////////////////////////////////////////////////
937             /*
                  ************************************************************************
                  *****************/
938             printf("start!¥n");
939             for ( i = 1; i <= Nz; i++ ) // z(i-1) ~ zi
940             {
```

```
941                 /* 相対遅延ステップ数の最大値算出 */
942                 Li =0;
943                 Li = (int) ( (( taumax - taumin )*( (double)i*dz ) / Tv) + 0.5 );
944
945                 /* 相対遅延ステップ数のモード分布算出 */
946                 for ( m = 0; m < NLP; m++ ) {
947                     kim[m] = (int) ( (( Mtau[m] - taumin )*( (double)i*dz ) / Tv ) +
                        0.5 )
948                                 - (int) ( (( Mtau[m] - taumin )*( (double)(i-1)
                        *dz ) / Tv ) + 0.5 ); }
949
950 #pragma omp parallel
951 {
952 #pragma omp for
953                 /* タイムシフト演算 */
954                 for ( m = 0; m < NLP; m++ ) {
955                     for ( n = 0; n < kim[m]; n++ ) { Amin[m][n] = 0.0; }
956                     for ( n = kim[m]; n <= Li; n++ ) { Amin[m][n] = Amplu[m][n - kim
                        [m]]; }}
957 #pragma omp for
958                 /* カップリング演算 */
959                 for ( m = 0; m < NLP; m++ ) {
960                     for ( n = 0; n <= Li; n++ ) { me=0.0;
961                     for ( l = 0; l < NLP; l++ ) { me = me + H[m][l]*Amin[l][n]; }
962                     Amplu[m][n] = me; }}
963 }
964
965                 // ① 波長分散を考慮する場合
966                 /* 時間波形の重ね合わせ */
967                 if ( matdis == 1 && i == Nz ) {
968                     // 各波長成分のインパルス応答（基準時間は非考慮）
969                     if ( y == Nl/2 || fout == 1) { fprintf ( fr2, "%f,", (double)
                        i*dz ); }
970                     for ( n = 0; n <= Li; n++ ) { ap = 0.0;
971                         for ( m = 0; m < NLP; m++ ) { ap = ap + Amplu[m][n]; }
972                         if ( y == Nl/2 || fout == 1) { fprintf ( fr2, "%f,", ap ); }}
973                     if ( y == Nl/2 || fout == 1) { fprintf ( fr2,"¥n" ); }
974                     // 最小群遅延の波長依存性を考慮した足し合わせ（基準時間を考慮）
975                     if ( y == 0 || nstd == 0 ) { for ( n = 0; n <= Li; n++ ) { for
                        ( m = 0; m < NLP; m++ ) { P[n] = P[n] + Amplu[m][n]; }}; }
976                     if ( y != 0 && nstd > 0 ) { for ( n = 0; n <= Li; n++ ) { for ( m
                        = 0; m < NLP; m++ ) { P[n+nstd] = P[n+nstd] + Amplu[m]
                        [n]; }}; }
977                     if ( y != 0 && nstd < 0 ) {
978                         if ( nstd < nstdmin ) {
979                             for ( n = 0; n <= Pnum[y-1] ; n++ ) { P[(Pnum[y-1]-n)
                        +(nstdmin-nstd)] = P[(Pnum[y-1]-n)]; }
980                             for ( n = 0; n < nstdmin-nstd; n++ ) { P[n] = 0.0; }
981                             for ( n = 0; n <= Li; n++ ) { for ( m = 0; m < NLP; m++ )
                        { P[n] = P[n] + Amplu[m][n]; }}}
982                         else {
983                             for ( n = 0; n <= Li; n++ ) { for ( m = 0; m < NLP; m++ )
                        {
984                                 P[n+(nstd-nstdmin)] = P[n+(nstd-nstdmin)] + Amplu[m]
                        [n]; }}; }};
985                     // for ( n = 0; n <= Pnum[y]; n++ ) { fprintf ( fs3,"%f,", P
                        [n] ); } fprintf ( fs3,"¥n" );
986                 }
987
988
989                 // ② 波長分散を考慮しない場合
990                 /* 計算結果の出力 */
991                 if ( matdis == 0 && (i%Nzout == 0 || i == Nz) ) {
992                     /* モードパワー分布 Pm */
993                     if ( fout == 1 || y == Nl/2 ) { fprintf ( fr3, "%f,", (double)
```

```
                    i*dz ); }
 994                for ( m = 0; m < NLP; m++ ) {
 995                    Pm[m] = 0.0;
 996                    for ( n = 0; n <= Li; n++ ) { Pm[m] = Pm[m] + Amplu[m][n]; }
 997                    if ( fout == 1 || y == NI/2 ) { fprintf ( fr3, "%f,", Pm
                    [m] ); }
 998                    Pg[modep[m]-1] = Pg[modep[m]-1] + Pm[m]; }
 999                if ( fout == 1 || y == NI/2 ) { fprintf ( fr3,"¥n"); }
1000                /* モード群パワー分布 Pg */
1001                if ( fout == 1 || y == NI/2 ) { fprintf ( fr9, "%f,", (double)
                    i*dz );
1002                for ( j = 0; j < Ptotal; j++ ) { fprintf ( fr9, "%f,", Pg[j] /
                    (double)pdeg[j] ); Pg[j] =0.0;}  fprintf ( fr9,"¥n"); }
1003                /* インパルス応答 P */
1004                if ( fout == 1 || y == NI/2) { fprintf ( fr2, "%f,", (double)
                    i*dz ); }
1005                for ( n = 0; n <= Li; n++ ) {
1006                    P[n] = 0.0;
1007                    for ( m = 0; m < NLP; m++ ) { P[n] = P[n] + Amplu[m][n]; }

1008                    if ( fout == 1 || y == NI/2 ) { fprintf ( fr2, "%f,", P
                    [n] ); } }
1009                if ( fout == 1 || y == NI/2 ) { fprintf ( fr2,"¥n" ); }

1010                /* 出力波形 */
1011                Pout = drealvector ( 0, Ti+Li ); init_realvector ( Pout, 0, Ti
                    +Li );
1012                Pout = convolution ( Ti, Pin, Li, P );
1013                if ( fout == 1 || y == NI/2) {
1014                    for ( n = 0; n < Ti+Li; n++ ) { fprintf ( fr6, "%f,", Pout
                    [n] );} fprintf ( fr6,"¥n" );}
1015                /* 周波数応答 M */
1016                if ( fout == 1 || y == NI/2 ) { fprintf ( fr4, "%f,", (double)
                    i*dz ); }
1017                for ( j = 0; j <= Nf; j++ ) {
1018                    Hw = ReHw = ImHw = 0.0;
1019                    for ( n = 0; n <= Li; n++ ) {
1020                        ReHw = ReHw + P[n]*cos ( (2.0*PI*(fmin+(double)j*df))*
                    (double)n*Tv );
1021                        ImHw = ImHw - P[n]*sin ( (2.0*PI*(fmin+(double)j*df))*
                    (double)n*Tv ); }
1022                    M[j] = sqrt ( ReHw*ReHw + ImHw*ImHw ) / A00; if ( y == NI/2
                    || fout == 1 ) { fprintf ( fr4, "%f,", -10.0*log10(1.0 / M
                    [j]) ); }}
1023                /* -3dB帯域幅 bw */
1024                bw = tav = Ptot = rms = 0.0;
1025                for ( j = 0; j <= Nf-1; j++ )  { if ( M[j] >0.5 && M[j+1] < 0.5 )
                     { bw = (fmin*1.0e3) + (df *1.0e3)*(j + (M[j] - 0.5) / (M[j] - M
                    [j+1])); break; } }
1026                if ( fout == 1 || y == NI/2 ) { fprintf ( fr4, "%f,", bw );}

1027                /* インパルス応答RMS幅 rms */
1028                for ( n = 0; n <= Li; n++ ) { tav = tav + ( taumin*(double)(i-1)
                    *dz + (double)n*Tv)*P[n]*Tv; Ptot = Ptot + P[n]*Tv; } tav =
                    tav / Ptot;
1029                for ( n = 0; n <= Li; n++ ) { rms = rms + ( taumin*(double)(i-1)
                    *dz + (double)n*Tv)*( taumin*(double)(i-1)*dz + (double)n*Tv)*
                    ( P[n] / Ptot )*Tv; }
1030                rms = sqrt ( rms - (tav*tav) );
1031
1032                if ( fout == 1 || y == NI/2 ) { fprintf ( fr4, "%f,%f", rms, tav
                    - ( taumin*(double)(i-1)*dz) ); fprintf ( fr4, "¥n" ); }
1033
1034                fprintf ( fs, "%f,%d,%d,%f,%f,%f,%f,%f,%f,%f¥n", g, NLP, NLP0,
                    taumin*(double)(i-1)*dz*1.0e-3  (double)i*dz  rms  bw  tav
```

```
                        Ptot, Aw00 );
1035                    printf ( "length:%f m ", (double)i*dz ); printf ( "-3db bandwidth:%f GHz ", bw ); printf ( "pulse broadening:%f ps¥n", rms );
1036                }
1037            }
1038            /*****************************************************************************/
1039
1040            if ( nstd < nstdmin ) { nstdmin = nstd; }
1041            if ( nstd > nstdmax ) { nstdmax = nstd; }
1042
1043            free_dmatrix ( H, 0, NLP-1, 0, NLP-1 );
1044            free_dmatrix ( Amin, 0, NLP-1, 0, Lmax );
1045            free_dmatrix ( Amplu, 0, NLP-1, 0, Lmax );
1046            free_drealvector ( alpha, 0 );
1047            free_dintvector ( kim, 0 );
1048            free_drealvector ( Pm, 0 );
1049            free_drealvector ( Pg, 0 );
1050            free_dintvector ( modem, 0 );
1051            free_dintvector ( model, 0 );
1052            free_dintvector ( modep, 0 );
1053            free_dintvector ( pdeg, 0 );
1054            free_drealvector ( Mbeta, 0 );
1055            free_drealvector ( Mtau, 0 );
1056            printf ( "A00=%f¥n", A00 );
1057            printf ( "End¥n¥n");
1058  } //! 波長ループ終了
1059  printf ( "Aw00=%f¥n", Aw00 );
1060
1061  //! ////////////////////////////////////////////////////
1062  //! ///////////          6. 結果の出力          ///////////
1063  //! ////////////////////////////////////////////////////
1064
1065  #if 1
1066  /* 計算結果の出力 */
1067  if ( matdis == 1 ) {
1068      /* インパルス応答波形（光源スペクトル考慮） Pw */
1069      for ( n = 0; n <= Pnum[NI]; n++ ) { fprintf ( fs3,"%f,", P[n] ); } fprintf ( fs3,"¥n" );
1070      /*  出力波形 */
1071      Pout = drealvector ( 0, Ti+Pnum[NI] ); init_realvector ( Pout, 0, Ti+Pnum[NI] );
1072      Pout = convolution ( Ti, Pin, Pnum[NI], P );
1073      if ( fout == 1 || y == NI/2) {
1074          for ( n = 0; n < Ti+Pnum[NI]; n++ ) { fprintf ( fs4, "%f,", Pout[n] );}
                fprintf ( fs4,"¥n" );}
1075      /* 周波数応答 M */
1076      fprintf ( fr4, "%f,", (double)Nz*dz );
1077      for ( j = 0; j <= Nf; j++ ) {
1078          ReHw = ImHw = 0.0;
1079          for ( n = 0; n <= Pnum[NI]; n++ ) {
1080              ReHw = ReHw + P[n]*cos ( (2.0*PI*(fmin+(double)j*df))*(double)n*Tv );
1081              ImHw = ImHw - P[n]*sin ( (2.0*PI*(fmin+(double)j*df))*(double)n*Tv ); }
1082          M[j] = sqrt ( ReHw*ReHw + ImHw*ImHw ) / Aw00; fprintf ( fr4, "%f,",
                -10.0*log10(1.0 / M[j]) ); }
1083
1084      /* -3dB帯域幅 bw */
1085      bw = tav = Ptot = rms = 0.0;
1086      for ( j = 0; j <= Nf-1; j++ )  { if ( M[j] >0.5 && M[j+1] < 0.5 ) { bw =
          (fmin*1.0e3) + (df *1.0e3)*(j + (M[j] - 0.5) / (M[j] - M[j+1])); break; }}
1087      fprintf ( fr4, "%f,", bw );
1088
1089      /* インパルス応答RMS幅 rms */
1090      for ( n = 0; n <= Pnum[NI]; n++ ) { tav = tav + ( taumin*(double)(i-1)*dz +
```

```
          (double)n*Tv)*P[n]*Tv; Ptot = Ptot + P[n]*Tv; } tav = tav / Ptot;
1091      for ( n = 0; n <= Pnum[Nl]; n++ ) { rms = rms + ( taumin*(double)(i-1)*dz +
            (double)n*Tv)*( taumin*(double)(i-1)*dz + (double)n*Tv )*( P[n] / Ptot )*Tv; }

1092      rms = sqrt ( rms - (tav*tav) );
1093      fprintf ( fr4, "%f,", rms); fprintf ( fr4, "¥n" );
1094
1095      fprintf ( fs, "%f,%f,%f,%f,%f,%f,%f,", g, (double)Nz*dz, rms, bw, tav, Ptot,
            AwOO );
1096      printf ( "length:%f m¥n", (double)Nz*dz ); printf ( "-3db bandwidth:%f GHz¥n",
            bw ); printf ( "rms width:%f ps¥n", rms ); }
1097
1098  #endif
1099
1100  /* スペックルコントラストの計算 */
1101  if ( scc == 1 ) {
1102      /* 光源スペクトル自己相関関数 Cp */
1103      Cp = drealvector ( 0, Nf ); init_realvector ( Cp, 0, Nf );
1104      for ( j = 0; j <= Nf; j++ ) {
1105          for ( i = 0; i <= Nfp; i++ ) {
1106              int jj = (int) ( (double)j*df / dfp );
1107              Cp[j] = Cp[j] + OSin[Nfp-i]*OSin[ Nfp-i+jj ]; }
1108          Cpsum = Cpsum + df*Cp[j]; }
1109
1110  // ガウス型スペクトル形状関数
1111  //      Cp[j] = Cp[j] + exp( -pow( (fpmin + dfp*double(i) - fp0) / fpgw, 2.0 ))
1112  //                              *exp( -pow( (fpmin + dfp*double(i) - df*double(j) -
        fp0 ) / fpgw, 2.0 ))*dfp; }
1113
1114      /* スペックルコントラスト spct */
1115      fprintf ( fr4, "," );
1116      for ( j = 0; j <= Nf; j++ ) {
1117          Cp[j] = Cp[j] / (2.0*Cpsum); fprintf ( fr4, "%f,", Cp[j] ); // Cpが偶関数であ
              ることを考慮
1118          spct = spct + Cp[j]*M[j]*M[j]*df; } spct = sqrt ( 2.0*spct );
1119          fprintf ( fs, "%f¥n", spct );   printf ( "speckle contrast:%f¥n", spct );
1120          free_drealvector (Cp, 0); }
1121
1122  #if 1
1123  ////printf("ifdis=1¥n");
1124  free_dintvector ( Pnum, 0 );
1125  free_drealvector ( P, 0 );
1126  free_drealvector ( Pin, 0 );
1127  free_drealvector ( Pout, 0 );
1128  free_drealvector ( M, 0 );
1129  #endif
1130
1131  fclose ( fp2 ); // BW_setting.csv
1132  fclose ( fp3 ); // BW_profile.csv
1133  fclose ( fp4 ); // FEM_result.csv
1134  fclose (fq);    // Hmn_result.csv
1135  fclose ( fr );  // CPE_Hmatrix.csv
1136  fclose ( fr2 ); // CPE_Impulse-responce.csv
1137  fclose ( fr3 ); // CPE_Mode-power-distribution.csv
1138  fclose ( fr4 ); // CPE_Frequency-response.csv
1139  fclose ( fr6 ); // CPE_Output-pulse-waveform.csv
1140  fclose (fs);    // BW_result.csv
1141  fclose (fs2);   // BW_Source-spectrum.csv
1142  fclose (fs3);   // BW_Impulse-responce.csv
1143  fclose (fs4);   // BW_Output-pulse-waveform.csv
1144
1145  system("pause");
1146  return 0;
1147  }
1148
1149  void selectOsciMode(char * directory) { //TODO ポインタ「*」は必要？
```

```
1150      FILE* fptr, *fptrcol;
1151      char fppath[128], s1[128], s2[65536];
1152
1153      int    m, l, NLP, NLP0, Ntotal, N, Nclad, Nbeta, mater, profile, Nwkb, Ntmin,
            Ntmax, Nf, launch, n;
1154      double delta, NA, aa, v, w, D;
1155      double tau, beta, dbeta, bb, eps1, eps2, sum, Rinf, de, df, eig, betaoverk;
1156      double taumin, taumax, fmin, fmax, dfrq, Hw, ReHw, ImHw, bw;
            //  int    nr;
1157      double yy, Rxy, Rinxy;  double ncav, noxi, Avin;
1158      double *Mbetain;        double** Rlp, ** MPD2d, ** Rinlp;;
1159      double tauin, betain, Rinfin, eigin, betain_devided_by_k, data;
1160      int    *modem, * modemin, * model, * modelin;
1161      int    Nvin, couple, min, lin, nrr1, nrr2, max;
1162      int    *m_guided, *l_guided;
1163      double *tau_guided, *beta_guided, * betaoverk_guided, * Rinf_guided, *
            eig_guided;
1164
1165      sprintf(fppath, "%s¥FEM_result_vcsel.csv", directory);
1166
1167      //! FEM_result_vcsel.csvの（列数・）行数の確認
1168      fptrcol = fopen(fppath, "r");
1169      n = -1;     //以下のwhileループでEOFを読みこんだ際，インクリメントされるため初期
            値で調整
1170      fscanf(fptrcol, "%d, %d, %lf, %lf, %lf, %lf, %lf,", &s1, &s1, &s1, &s1, &s1, &s1,
              &s1);
1171      //TODO 列数を以下でカウント
1172      /* for(){
1173      fscanf(fptrcol,
1174      }*/
1175      while (!feof(fptrcol) && n < 512) { n++;
1176          fscanf(fptrcol, "%[^¥n]", &s2); }
1177      printf("FEM_result_vcsel.csv の行数は%dです¥n", n);
1178      fclose(fptrcol);
1179      m_guided = dintvector(0, n);
1180
1181
1182      if ((fptr = fopen(fppath, "r")) != NULL) {
1183          for (int i = 0; i <= n; i++) {
1184              fscanf(fptr, "%d, %d, %lf, %lf, %lf, %lf, %lf,", &m_guided[i], &l_guided
                  [i], &tau_guided[i], &beta_guided[i], &betaoverk_guided[i],
                  &Rinf_guided[i], &eig_guided[i]);
1185              for (int j = 0; j <= N; j++) {
1186                  fscanf(fptr, "%lf,", R2[j]);
1187                  Rlp[NLP][j] = R2[j];
1188          }}
1189
1190          fprintf(fp4, "¥n");
1191
1192
1193
1194
1195          //printf("m, l, tau, beta, ne, eig, R[N]¥n");
1196          //fprintf(fp4, "m,l,tau,beta,ne,R_infinite,eig,r=0_um,");
1197          //for (j = 1; j <= N; j++) { fprintf(fp4, "%lf,", (double)j * dr * 1.0e6); }
              fprintf(fp4, "¥n");
1198      }
1199      else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1200
1201
1202  }
1203
1204  char inputFEM() {
1205      FILE   *fp, *fp2, *fp3, *fp4, *fp5, *fp6, *fp7, *fp8, *fr, *fpulse, *fopulse,
              *fmpd;
1206      char   fppath[128], fp2path[128], fp3path[128], fp4path[128], fp5path[128],
              fp6path[128], fp7path[128];
1207      char   fp8path[128], frpath[128], fpulsepath[128], fopulsepath[128], fmpdpath
```

```
        [128];
1208            fppath[0]   =  fp2path[0] = fp3path[0] = fp4path[0] = fp5path[0] =
                  fp6path[0] = fp7path[0] =
1209            fp8path[0]  =  frpath[0] = fpulsepath[0] = fopulsepath[0] = fmpdpath[0]
                  = '¥0' ;
1210    int    myu, x, y, i, j, jmax, count;
1211    int    m, l, NLP, NLP0, Ntotal, N, Nclad, Nbeta, mater, profile, Nwkb, Ntmin,
          Ntmax, Nf, launch;
1212    int    Nxy, xmax, LT, xL;
1213    double lamda, k, omega, A, AA, g, n0, n1, dr, L, Tv;
1214    double r0, w0, dx, dy;
1215    double delta, NA, aa, v, w, D;
1216    double tau, beta, dbeta, bb, eps1, eps2, sum, Rinf, de, df, eig;
1217    double taumin, taumax, fmin, fmax, dfrq, Hw, ReHw, ImHw, bw;
          //  int    nr;
1218    double yy, Rxy, Rinxy;
1219    double Em_evin, Em_odin, Em_ev, Em_od, Am_evev, Am_evod, Am_odev, Am_odod;
1220    double ncav, noxi, Avin;
1221    double *GI, *pulse, *opulse, *q, *qg, *R, *R2, *Rb, *a, *b, *ML, *MD, *Mtau, *P,
          *M, *Mbeta, *MPD;
1222    double *Mbetain;
1223    double **Rlp, **MPD2d, **Rinlp;;
1224    double tauin, betain, Rinfin, eigin, betain_devided_by_k, data;
1225    double win, xx1, xx2, rr1, rr2;
1226    int    *modem, *modemin, *model, *modelin;
1227    int    Nvin, couple, min, lin, nrr1, nrr2, max;
1228    int    OFFres, OFFrange;
1229    char   trash[65536] = "¥0";
1230    char   directory[128] = "FILE_for_IO";
1231    mkdir(directory);
1232    /** 1. 入力ファイルの読み込み **/
1233    if ((fp = fopen("[FEM_VCSEL].csv", "r")) != NULL) {
1234        char s1[256], s2[256];
1235        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &A);
1236        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &AA);
1237        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &profile);
1238        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &g);
1239        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &n1);
1240        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &n0);
1241        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &launch);
1242        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &couple);
1243        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &r0);
1244        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &OFFrange);
1245        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &OFFres);
1246        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &Avin);
1247        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &w0);          // ガウス分布（シング
          ルモードレーザ）を用いて励振
1248        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &dr);
1249        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &dx);
1250        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &mater);
1251        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &lamda);
1252        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &L);
1253        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &Tv);
1254        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &fmin);
1255        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &fmax);
1256        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &Nf);
1257        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &aa);
1258        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &Nbeta);
1259        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &eps1);
1260        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &eps2);
1261        fscanf(fp, "%[^,], %[^,], %d¥n", s1, s2, &jmax);
1262        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &noxi);
1263        fscanf(fp, "%[^,], %[^,], %lf¥n", s1, s2, &ncav);
1264        // fscanf(fp, "%[^,]¥n", s1);
```

```
1265        }
1266        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1267
1268        N = (int)(1000.0*A / dr); Nclad = (int)(1000.0*(AA - A) / dr);                    //----------Ncladに注意
1269        Nvin = (int)(1000.0*Avin / dr);
1270
1271
1272        fmin = fmin * 1.0e-3; fmax = fmax * 1.0e-3; dfrq = (fmax - fmin) / (double)Nf;
1273
1274        /************************    ここが非常に重要    ***************************/
1275        Nxy = (int)(3 * 1000 * A / dx); //限定モード励振時の解析領域に対応する分割数    Nxy*dx=解析領域
1276        /*****************************************************************************/
1277
1278        lamda = lamda * 1.0e-9;  A = A * 1.0e-6; dr = dr * 1.0e-9; AA = AA * 1.0e-6;
1279        Avin = Avin * 1.0e-6;
1280        r0 = r0 * 1.0e-6; w0 = w0 * 1.0e-6, dx = dx * 1.0e-9; dy = dx;  // 単位変換 (THz (1/ps))
1281        fclose(fp);
1282
1283        /* 測定プロファイル読み込み */
1284        GI = drealvector(0, N); init_realvector(GI, 0, N);
1285        sprintf(frpath, "%s/profile.csv", directory);
1286        if (profile == 1) {
1287            if ((fr = fopen(frpath, "r")) != NULL) {
1288                for (j = 0; j <= N; j++) { fscanf(fr, "%lf,", &GI[j]); }
1289            }
1290            else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1291            n1 = GI[0]; n0 = GI[N]; fclose(fr);
1292        }
1293
1294        /* 入射波形読み込み */
1295        pulse = drealvector(0, 1999); init_realvector(pulse, 0, 1999); //入射波形は2000ステップで入力
1296        sprintf(fpulsepath, "%s/input pulse.csv", directory);
1297        if ((fpulse = fopen(fpulsepath, "r")) != NULL) {
1298            for (j = 0; j <= 1999; j++) { fscanf(fpulse, "%lf,", &pulse[j]); }
1299        }
1300        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1301        fclose(fpulse);
1302
1303
1304        /** 2. 各種定数の設定 **/
1305        k = 2.0*PI / lamda; omega = 2.0*PI*C / lamda;
1306        delta = (n1*n1 - n0 * n0) / (2.0*n1*n1); NA = sqrt(n1*n1 - n0 * n0);
1307        v = k * aa*n1*sqrt(2.0*delta); D = A / aa;
1308
1309        Nwkb = int((1.0 / 4.0)*(g / (g + 2.0))*(k*k)*(n1*n1)*delta*(A*A));
1310        bb = -1; dbeta = k * (n1 - n0) / (double)Nbeta;//初期化
1311        NLP = NLP0 = Ntotal = 0;//初期化
1312
1313                                /* 配列の記憶領域確保および初期化 */
1314        q = drealvector(0, N); init_realvector(q, 0, N);
1315        qg = drealvector(0, N); init_realvector(qg, 0, N);
1316        R = drealvector(0, N); init_realvector(R, 0, N);
1317        R2 = drealvector(0, N); init_realvector(R2, 0, N);
1318        Rb = drealvector(0, N); init_realvector(Rb, 0, N);
1319        a = drealvector(0, N); init_realvector(a, 0, N);
1320        b = drealvector(0, N); init_realvector(b, 0, N);
1321        ML = drealvector(0, N); init_realvector(ML, 0, N);
1322        MD = drealvector(0, N); init_realvector(MD, 0, N);
1323        M = drealvector(0, Nf); init_realvector(M, 0, Nf);
1324        Mtau = drealvector(0, 10 * Nwkb); init_realvector(Mtau, 0, 10 * Nwkb);
1325        modem = dintvector(0, 10 * Nwkb); init_intvector(modem, 0, 10 * Nwkb);
1326        model = dintvector(0, 10 * Nwkb); init_intvector(model, 0, 10 * Nwkb);
1327        modelin = dintvector(0, 10 * Nwkb); init_intvector(modelin, 0, 10 * Nwkb);
1328        Mbeta = drealvector(0, 10 * Nwkb); init_realvector(Mbeta, 0, 10 * Nwkb);
```

```
1329        MPD = drealvector(0, 10 * Nwkb); init_realvector(MPD, 0, 10 * Nwkb);
1330
1331        #define sizeofRLP 500
1332        RIp = dmatrix(0, sizeofRLP, 0, sizeofRLP);
1333        for (int i = 0; i <= sizeofRLP; i++) {
1334            for (int j = 0; j <= sizeofRLP; j++) { RIp[i][j] = 0.0; }
1335        }
1336
1337        /* 屈折率分布の設定 */
1338        /* べき乗プロファイル*/
1339        if (profile == 0) {
1340            for (j = 0; j <= N; j++) { GI[j] = n1 * sqrt(1.0 - 2.0*delta*pow(((double)j /
                (double)N), g)); }
1341        }
1342        for (j = 0; j <= N; j++) { q[j] = (GI[j] * GI[j] - n0 * n0) / (n1*n1 - n0 *
              n0); }
1343        for (j = 0; j <= N; j++) { qg[j] = GI[j] * GI[j] - (lamda*GI[j] * dndl
              (lamda*1.0e9, GI[j], mater)) / (1 - (lamda / GI[j])*dndl(lamda*1.0e9, GI[j],
              mater)); }
1344
1345
1346        /** 3. 評価条件の出力 **/
1347        sprintf(fp2path, "%s/FEM_setting.csv", directory);
1348        if ((fp2 = fopen(fp2path, "w")) != NULL) {
1349            fprintf(fp2, "Material, mater, %d¥n", mater);
1350            fprintf(fp2, "Wavelength, λ, %lf,nm¥n", lamda*1e9);
1351            fprintf(fp2, "Fiber length, L, %lf,m¥n", L);
1352            fprintf(fp2, "Core radius, A, %lf,μm¥n", A*1e6);
1353            fprintf(fp2, "Analysis region in radial axis, AA,%lf,μm¥n", AA*1e6);
1354            fprintf(fp2, "Index exponent, g, %lf¥n", g);
1355            fprintf(fp2, "Refractive index at the core center, n1,%lf¥n", n1);
1356            fprintf(fp2, "Refractive index in the cladding, n0,%lf¥n", n0);
1357            fprintf(fp2, "Relative refractive index, Δ,%lf¥n", delta);
1358            fprintf(fp2, "Numerical aperture, NA, %lf¥n", NA);
1359            fprintf(fp2, "Step size of the elements, dr, %lf, nm¥n", dr*1e9);
1360            fprintf(fp2, "Step size of the pulse waveform, Tv, %lf, ps¥n", Tv);
1361            fprintf(fp2, "Minimum evaluated frequency,fminx,%e,GHz¥n", fmin*1.0e3);
1362            fprintf(fp2, "Maximum evaluated frequency,fmax,%e,GHz¥n", fmax*1.0e3);
1363            fprintf(fp2, "Step size of evaluated frequency,dfrq,%e,GHz¥n", dfrq*1.0e3);
1364            fprintf(fp2, "Step size of propagation constant, dβ,%lf¥n", dbeta);
1365            fprintf(fp2, "Partition number of propagation constant, Nβ,%d¥n", Nbeta);
1366            fprintf(fp2, "Partition number of fiber core radius, N,%d¥n", N);
1367            fprintf(fp2, "Partition number of fiber cladding, Nclad,%d¥n", Nclad);
1368            fprintf(fp2, "Maximum allowable error for convergence solution vector, eps1,%
                lf¥n", eps1);
1369            fprintf(fp2, "Maximum allowable error of zero eigen value, eps2,%lf¥n",
                eps2);
1370            fprintf(fp2, "Maximum number of iterations in inverse power method, jmax,%d
                ¥n", jmax);
1371            printf("Material no.: %d¥n", mater);
1372            printf("Wavelength λ: %lfnm¥n", lamda*1.0e9);
1373            printf("Fiber length L: %lfm¥n", L);
1374            printf("Core radius A: %lfμm¥n", A*1.0e6);
1375            printf("Analysis region AA: %lf μm¥n", AA*1e6);
1376            printf("Index exponent g: %lf¥n", g);
1377            printf("Refractive index at the core center n1: %lf¥n", n1);
1378            printf("Refractive index in the cladding n0: %lf¥n", n0);
1379            printf("Relative refractive index Δ: %lf¥n", delta);
1380            printf("Numerical aperture NA: %lf¥n", NA);
1381            printf("Step size of the elements dr: %lfnm¥n", dr*1.0e9);
1382            printf("Partition number of fiber core radius N: %d¥n", N);
1383            printf("Partition number of fiber cladding Nclad: %d¥n", Nclad);
1384            printf("Step size of propagation constants dβ: %lf¥n", dbeta);
1385            printf("Partition number of propagation constants Nβ: %d¥n", Nbeta);
1386            printf("Maximum allowable error for convergence solution vector eps1: %lf¥n",
                eps1);
```

```
1387            printf("Maximum allowable error of zero eigen value eps2: %lf¥n", eps2);
1388            printf("Maximum number of iterations in inverse power method, jmax,%d¥n",
                  jmax);
1389        }
1390        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1391
1392        sprintf(fp3path, "%s/FEM_profile.csv", directory);
1393        if ((fp3 = fopen(fp3path, "w")) != NULL) {
1394            fprintf(fp3, "r [μm], n, q, qg¥n");
1395            for (j = 0; j <= N; j++) {
1396                fprintf(fp3, "%lf, %lf, %lf, %lf¥n", A*1.0e6* (double)j / (double)N, GI
                      [j], q[j], qg[j]);
1397            }
1398        }
1399        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1400        fclose(fp3);
1401
1402        sprintf(fp4path, "%s/FEM_result_vcsel.csv", directory);
1403        if ((fp4 = fopen(fp4path, "w")) != NULL) {
1404            printf("m, l, tau, beta, ne, eig, R[N]¥n");
1405            fprintf(fp4, "m,l,tau,beta,ne,R_infinite,eig,r=0_um,");
1406            for (j = 1; j <= N; j++) { fprintf(fp4, "%lf,", (double)j*dr*1.0e6); }
                  fprintf(fp4, "¥n");
1407        }
1408        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1409
1410        sprintf(fp5path, "%s/FEM_impulse_response.csv", directory);
1411        if ((fp5 = fopen(fp5path, "w")) != NULL) { ; }
1412        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1413
1414        sprintf(fp6path, "%s/FEM_frequency-respnse.csv", directory);
1415        if ((fp6 = fopen(fp6path, "w")) != NULL) {
1416            for (y = 0; y <= Nf; y++) { fprintf(fp6, "%lf,", (fmin*1.0e3) + (double)y*
                  (dfrq*1.0e3)); } fprintf(fp6, "¥n");
1417        }
1418        else { printf("The file cannot be opened !¥n"); exit(EXIT_FAILURE); }
1419
1420        sprintf(fopulsepath, "%s/FEM_output pulse.csv", directory);
1421        if ((fopulse = fopen(fopulsepath, "w")) != NULL) { ; }
1422        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1423
1424        sprintf(fmpdpath, "%s/FEM_mode power distribution.csv", directory);
1425        if ((fmpd = fopen(fmpdpath, "w")) != NULL) { ; }
1426        else { printf(" U cannot open the file !¥n"); exit(EXIT_FAILURE); }
1427
1428
1429        /** 4. 固有ベクトルおよび固有値の計算 **/
1430        for (m = 0; ; m++) {
1431            l = 1; beta = k * n1;
1432
1433            /*
                  ********************************************************************************
                  ************/
1434            for (; ; ) {
1435                /* 係数行列計算および改訂コレスキー分解 */
1436                w = aa * sqrt((beta*beta) - (k*k)*(n0*n0));
1437                S_matrix(a, b, q, m, N, v, w, D);
1438                mcholesky(a, b, ML, MD, m, N);
1439                /* 初期ベクトル R0 の付与 */
1440                R0(MD, R, m, N);
1441                /*  連立一次方程式 SR=(LDL)R=bR の反復評価 */
1442                for (j = 0; ; j++) {
1443                    for (i = 0; i <= N; i++) { Rb[i] = R[i]; }
1444                    mcholesky_sol(ML, MD, R, m, N);
1445                    R_norm(R, N);
1446                    /* 収束判定 */
1447                    de = 0, df = 0;
```

```
1448                    for (i = 0; i <= N; i++) {
1449                        de = de + (Rb[i] - R[i])*(Rb[i] - R[i]);
1450                        df = df + (Rb[i] + R[i])*(Rb[i] + R[i]);
1451                    }
1452
1453                    if (de < eps1 || df < eps1) break;
1454                    if (j >= jmax) goto nextin;
1455                    // ① RとRbの成分差deが0に漸近すれば収束（break）.
1456                    // ② RとRbの成分和dfが0に漸近すれば中止（break）.
1457                    // ③ 反復回数が上限値 jmax を超えたらβを変更して再計算（go to
                          next）.
1458                }
1459
1460                /* 固有値の計算 */
1461                eig = Eigen(R, a, b, N, m);
1462
1463                /* 固有値の妥当性評価 */
1464                // 「収束固有値 eig が前回値 bb と同値」であればβを変えて初めから再計算
1465                if (eig == bb) {
1466                    dbeta = k * (n1 - n0) / (double)Nbeta;
1467                    beta = beta - 1.0*dbeta;
1468                    continue;
1469                }
1470
1471                //  ①「0< 収束固有値 eig < 0.00001」であれば零固有値として採用
1472                if (0.0 < eig && eig < eps2) {
1473                    /* 横方向電場成分Rの規格化 （パワーを1Wとする）*/
1474                    sum = 0.0;
1475                    for (j = 0; j < N; j++) { sum = sum + R[j] * R[j] * (j*dr)*dr; }
1476                    for (j = 0; j < Nclad; j++) { sum = sum + R[N] * (bessk(m, w*(j*dr +
                          A)) / bessk(m, w*A))*R[N] * (bessk(m, w*(j*dr + A)) / bessk(m,
                          w*A))*(j*dr + A)*dr; }
1477                    for (j = 0; j <= N; j++) { R2[j] = R[j] * sqrt((omega*Mu0) /
                          (PI*beta*sum)); }
1478                    tau = Mtau[NLP] = (1.0 / (C*1.0e-12))*(k / beta) * dbdk_bunshi(R, qg,
                          D, w, m, N) / dbdk_bunbo(R, D, w, m, N); // = (1/c)*(dβ/dk) [ps/
                          m]
1479                    modem[NLP] = m;          model[NLP] = l;
1480                    Mbeta[NLP] = beta;
1481
1482                    /* 計算結果の出力 */
1483                    Rinf = R[N] * (bessk(m, w*(Nclad*dr + A)) / bessk(m, w*A));
1484                    fprintf(fp4, "%d, %d, %lf, %lf, %lf, %lf, %lf,", m, l, tau, beta,
                          beta / k, Rinf, eig);
1485                    for (j = 0; j <= N; j++) {
1486                        fprintf(fp4, "%lf,", R2[j]);
1487                        Rlp[NLP][j] = R2[j];
1488                    }
1489
1490                    fprintf(fp4, "¥n");
1491                    printf("%d, %d, %lf, %lf, %lf, %lf, %lf¥n", m, l, tau, beta, beta /
                          k, eig, R[N]);
1492                    dbeta = k * (n1 - n0) / (double)Nbeta;
1493                    bb = eig;
1494                    l = l + 1;
1495                    NLP = NLP + 1; if (m == 0) { NLP0 = NLP0 + 1; }
1496                    count = 0;
1497                }
1498
1499                //  ②「0 < 収束固有値 eig」かつ「-1 < 前回値 bb < 0」であれば
1500                if (eig > 0.0 && bb < 0.0 && (fabs(bb) < 1.0)) {
1501                    beta = beta + dbeta;
1502                    dbeta = dbeta / 2.0;
1503                    count = count + 1;
1504                }
1505
1506                // ③ その他
```

```
1507                else { bb = eig; count = 0; }
1508                if (count > 3000) {
1509                    dbeta = k * (n1 - n0) / (double)Nbeta;
1510                    beta = beta - dbeta;
1511                }
1512
1513            nextin:
1514                beta = beta - dbeta;
1515                if (beta < k*n0)    break;
1516            }
1517            /*************************************************************************
                ************/
1518            if (l == 1) { Ntotal = (NLP0) * 2 + (NLP - NLP0) * 4; break; }
1519        }       // breakとなるのは, mが最高次数に到達したとき
1520        printf("固有値計算完了¥n");
1521        free_drealvector(GI, 0);
1522        free_drealvector(q, 0);
1523        free_drealvector(qg, 0);
1524        free_drealvector(R, 0);
1525        free_drealvector(R2, 0);
1526        free_drealvector(Rb, 0);
1527        free_drealvector(a, 0);
1528        free_drealvector(b, 0);
1529        free_drealvector(ML, 0);
1530        free_drealvector(MD, 0);
1531        free_drealvector(Mtau, 0);
1532        free_dintvector(modem, 0);
1533        //free_drealvector(P, 0);
1534        free_drealvector(M, 0);
1535        //
1536        free_drealvector(Mbeta, 0);
1537        free_drealvector(MPD, 0);
1538        free_drealvector(pulse, 0);
1539        //free_drealvector(opulse, 0);
1540
1541        fprintf(fp2, "Total number of LP modes (WKB), Nwkb,%d¥n", Nwkb);
1542        fprintf(fp2, "Total number of LP modes, NLP,%d¥n", NLP);
1543        fprintf(fp2, "Total number of LP modes with 0th m-order, NLP0,%d¥n", NLP0);
1544        fprintf(fp2, "Total number of modes, Ntotal,%d¥n", Ntotal);
1545
1546        fclose(fp2);
1547        fclose(fp4);
1548        fclose(fp5);
1549        fclose(fp6);    //fclose(fp7);
1550        fclose(fopulse);
1551        fclose(fmpd);
1552
1553        return directory[128];
1554    }
1555
1556
1557    //! //////////      以下サブルーチン       //////////
1558    //
1559    /*A.1. 第2種変形ベッセル関数 bessk (n, x) */
1560    // 第1種変形Bessel関数 (n=0) I0(x)
1561    double bessi0 (double x)
1562    {
1563        double ax, ans;
1564        double y;
1565        // Polynomial fit
1566        if ( ( ax = fabs(x) ) < 3.75 ) {
1567            y = x / 3.75;
1568            y*= y;
1569            ans = 1.0 + y*(3.5156229 + y*(3.0899424 + y*(1.2067492
1570                + y*(0.2659732 + y*(0.360768e-1+y*0.45813e-2)))));}
1571        else {
```

```
1572            y = 3.75 / ax;
1573            ans=(exp(ax) / sqrt(ax))*(0.39894228 + y*(0.1328592e-1
1574                +y*(0.225319e-2 + y*(-0.157565e-2 + y*(0.916281e-2
1575                +y*(-0.2057706e-1 + y*(0.2635537e-1 + y*(-0.1647633e-1
1576                +y*0.392377e-2))))))));}
1577        return ans;
1578    }
1579    // 第2種変形Bessel関数 (n=0)  K0(x)
1580    double bessk0 (double x)
1581    {
1582        double bessi0 (double x);
1583        double y, ans;
1584        //polynomial fit
1585        if ( x <= 2 ) {
1586            y = x*x / 4.0;
1587            ans = (-log(x/2.0)*bessi0(x)) + (-0.57721566 + y*(0.42278420
1588                + y*(0.23069756 + y*(0.3488590e-1 + y*(0.262698e-2
1589                + y*(0.10750e-3 + y*0.74e-5))))));
1590        }
1591        else {
1592            y=2.0/x;
1593            ans = (exp(-x)/sqrt(x))*(1.25331414 + y*(-0.7832358e-1
1594                + y*(0.2189568e-1 + y*(-0.1062446e-1 + y*(0.587872e-2
1595                + y*(-0.251540e-2 + y*0.53208e-3))))));
1596        }
1597        return ans;
1598    }
1599    // 第1種変形Bessel関数 (n=1)  I1(x)
1600    double bessi1 (double x)
1601    {
1602        double ax, ans;
1603        double y;
1604        if ( ( ax = fabs(x) ) < 3.75 ) {
1605            y = x / 3.75;
1606            y*=y;
1607            ans = ax*(0.5 + y*(0.87890594 + y*(0.51498869 + y*(0.15084934
1608                + y*(0.2658733e-1 + y*(0.301532e-2 + y*0.32411e-3))))));}
1609        else {
1610            y = 3.75 / ax;
1611            ans = 0.2282967e-1 + y*(-0.2895312e-1 + y*(0.1787654e-1
1612                - y*0.420059e-2));
1613            ans = 0.39894228 + y*(-0.3988024e-1 + y*(-0.362018e-2
1614                + y*(0.163801e-2 + y*(-0.1031555e-1 + y*ans))));
1615            ans*=(exp(ax) / sqrt(ax));
1616        }
1617        return x < 0.0 ? - ans : ans;
1618    }
1619    // 第2種変形Bessel関数 (n=1)  K1 (x)
1620    double bessk1 (double x) {
1621        double bessi1 (double x);
1622        double y, ans;
1623        if ( x <= 2.0 ) {
1624            y = x*x/4.0;
1625            ans = (log(x/2.0)*bessi1(x)) + (1.0/x)*(1.0 + y*(0.15443144
1626                + y*(-0.67278579 + y*(-0.18156897 + y*(-0.1919402e-1
1627                + y*(-0.110404e-2 + y*(-0.4686e-4)))))));}
1628        else{
1629            y = 2.0/x;
1630            ans = (exp(-x)/sqrt(x))*(1.25331414 + y*(0.23498619
1631                + y*(-0.3655620e-1 + y*(0.1504268e-1 + y*(-0.780353e-2
1632                + y*(0.325614e-2 + y*(-0.68245e-3)))))));
1633        }
1634        return ans;
1635    }
1636    // 第2種変形Bessel関数 Kn(x)
1637    double bessk (int n,double x)
1638    {
```

```
1639      double bessnorm = 1.0e7 ;
1640      double bessk0 (double x);
1641      double bessk1 (double x);
1642      int j;
1643      double bk,bkm,bkp,tox;
1644      if ( n == 0 ) return bessk0 (x) / bessnorm;
1645      if ( n == 1 ) return bessk1 (x) / bessnorm;
1646      if ( n >= 2 ) {
1647          tox = 2.0/x;
1648          bkm = bessk0(x) / bessnorm;
1649          bk = bessk1(x) / bessnorm;
1650          for ( j=1; j<n; j++ ) {
1651              bkp = bkm + j*tox*bk;
1652              bkm = bk;
1653              bk = bkp;
1654          }
1655          return bk;
1656      }
1657      else return 0;
1658  }
1659
1660  /*A.2. 屈折率波長微分関数dndl_var (lamda, n_lamda, mater) */
1661  double dndl ( double lamda, double n_lamda, int mater )
1662  //{ return ( ( 0.01925e-4*lamda - 16.31619e-4 )*n_lamda + ( -0.02743e-4*lamda +
      23.16674e-4 ) )*1.0e9; }
1663  { return ( ( 0.02173e-4*lamda - 18.79107e-4 )*n_lamda + ( -0.03109e-4*lamda +
      26.85035e-4 ) )*1.0e9; } // 640 ~ 690 nm
1664
1665  /*A.3. 屈折率濃度微分関数dndl_var (lamda, mater) */
1666  double dndc ( double lamda, int mater )
1667  { return 2.03716e-9*lamda*lamda - 3.27125e-6*lamda + 2.98314e-3; } // 589 ~ 690 nm
1668
1669  /*A.3. コア中心屈折率 ncore (lamda, mater) */
1670  double ncore ( double lamda, int mater ) {
1671      double sell; sell = 1.0;
1672      /* DPS-doped PMMA ( 7.8 wt.%, 1.506@589nm ) */
1673      sell = sqrt ( 1.0 + ( 0.41241 / (1.0 - ( 22500 / (lamda*lamda) )) )
1674          + ( 0.81215 / (1.0 - ( 6400 / (lamda*lamda) )) )
1675          + ( 0.01117 / (1.0 - ( 11560000 / (lamda*lamda) )) ) );
1676      /* DPS-doped PMMA ( 9.0 wt.%, 1.506@655nm ) */
1677  //  sell = sqrt ( 1.0 + ( 0.61249 / (1.0 - ( 8467.04111 / (lamda*lamda) )) )
1678  //      + ( 0.61872 / (1.0 - ( 16525.45233 / (lamda*lamda) )) )
1679  //      + ( 0.08889 / (1.0 - ( 129601870.30589 / (lamda*lamda) )) ) );
1680      return sell; }
1681
1682  /*A.4. クラッド屈折率 nclad (lamda, mater) */
1683  double nclad ( double lamda, int mater ) {
1684      double sell; sell =1.0;
1685      /* PMMA ( 1.492@589nm ) */
1686      sell = sqrt ( 1.0 + ( 0.496284 / (1.0 - ( 5154.872 / (lamda*lamda) )) )
1687          + ( 0.6964977 / (1.0 - ( 13802.53 / (lamda*lamda) )) )
1688          + ( 0.3223 / (1.0 - ( 85527690 / (lamda*lamda) )) ) );
1689      /* DPS-doped PMMA ( 1.079427062 wt. % ) */
1690  //  sell = sqrt ( 1.0 + ( 0.5954 / (1.0 - ( 6200.94518 / (lamda*lamda) )) )
1691  //      + ( 0.602 / (1.0 - ( 14730.91236 / (lamda*lamda) )) )
1692  //      + ( 0.29126 / (1.0 - ( 85685787.87303 / (lamda*lamda) )) ) );
1693      return sell; }
1694
1695  /*A.5. 係数行列要素計算関数 S_matrix (a, b, q, m, n, v, w, D)*/
1696  // b[0]=S00, b[1]=S11, ..... b[j]=Sjj ..... b[n]=Snn
1697  // a[0]=___, a[1]=S01, a[2]=S12, ... a[j]=Sj-1,j ... a[n]=Sn-1,n
1698  // a[0]=___, a[1]=S10, a[2]=S21, ... a[j]=Sj,j-1 ... a[n]=Sn,n-1
1699  void S_matrix ( double *a, double *b, double *q, int m, int n, double v, double w,
      double D )
1700  {
1701      int j;
```

```
1702      if ( m == 0 ) {
1703              b[0] = - (1.0/2.0) + (3.0*q[0]+2.0*q[1])*(v*v/60.0)*((D*D)/(n*n)) -
                    (1.0/12.0)*(w*w)*((D*D)/(n*n));
1704              b[n] = - ((2.0*n-1.0)/2.0) + ((5.0*n-2.0)*q[n-1]+3.0*(5.0*n-1.0)*q[n])*
                    (v*v/60.0)*((D*D)/(n*n)) - ((4.0*n-1.0)/12.0)*(w*w)*((D*D)/(n*n))
1705                        - w*D*bessk(1,w*D)/bessk(0,w*D);}
1706      else {
1707              b[0] = 0.0;
1708              b[n] = - (1.0-m*m)*((2.0*n-1.0)/2.0) + ((5.0*n-2.0)*q[n-1]+3.0*
                    (5.0*n-1.0)*q[n])*(v*v/60.0)*((D*D)/(n*n)) - ((4.0*n-1.0)/12.0)*(w*w)*
                    ((D*D)/(n*n))
1709                        - (m*m)*((n-1.0)*(n-1.0))*log((double)n/((double)n-1.0)) - m*m
                        - w*D*bessk(m-1,w*D)/bessk(m,w*D) - m ; }
1710
1711              b[1] = - 2.0*(1.0-m*m) + (3.0*q[0]+30*q[1]+7.0*q[2])*(v*v/60.0)*((D*D)/
                    (n*n)) - (2.0/3.0)*(w*w)*((D*D)/(n*n)) - (m*m)*4.0*log(2.0);
1712          for ( j = 2; j < n; j++ )
1713          {   b[j] = - 2.0*j*(1.0-m*m) + ((5.0*j-2.0)*q[j-1]+30.0*j*q[j]+(5.0*j+2.0)*q
                [j+1])*(v*v/60.0)*((D*D)/(n*n)) - (2.0/3.0)*j*(w*w)*((D*D)/(n*n))
1714                        - (m*m)*( ((j-1.0)*(j-1.0))*log((double)j/((double)j-1.0)) +
                        ((j+1.0)*(j+1.0))*log( ((double)j+1.0) / (double)j) ); }
1715
1716              a[0] = 0.0;//未使用要素につき0を格納
1717              a[1] = (1.0/2.0) + (2*q[0]+3*q[1])*(v*v/60.0)*((D*D)/(n*n)) - (1.0/12.0)*
                    (w*w)*((D*D)/(n*n)) - (m*m)/2;
1718          for ( j = 2; j <= n; j++ )
1719          {   a[j] = ((2.0*j-1.0)/2.0)*(1.0-m*m) + ((5.0*j-3.0)*q[j-1] + (5.0*j-2.0)*q
                [j])*(v*v/60.0)*((D*D)/(n*n)) - ((2.0*j-1.0)/12.0)*(w*w)*((D*D)/(n*n))
1720          + (m*m)*(j-1.0)*j*log((double)j / ((double)j-1.0)); }
1721  }
1722
1723  /*A.4. 改訂コレスキー分解   mcholesky ( a, b, ML, MD, m, n )*/
1724  void mcholesky ( double *a, double *b, double *ML, double *MD, int m, int n )
1725  {
1726      int i;
1727      if ( m == 0 ) {
1728          ML[0] = 0.0;//未使用要素につき0を格納
1729          MD[0] = b[0];
1730          for ( i = 1; i <= n; i++ )
1731          {   MD[i] = b[i] - a[i]*a[i] / MD[i-1];
1732              ML[i] = a[i] / MD[i-1]; }
1733      }
1734      else {
1735          ML[0] = 0.0;//未使用要素につき0を格納
1736          ML[1] = 0.0;//発散するから別扱い. 0で良いのか?
1737          MD[0] = 0.0;
1738          MD[1] = b[1];
1739          for ( i = 2; i <= n; i++ )
1740          {   MD[i] = b[i] - a[i]*a[i] / MD[i-1];
1741              ML[i] = a[i] / MD[i-1]; }
1742      }
1743  }
1744
1745  /*A.5. 改訂コレスキー分解法により方程式を解く   mcholesky_sol ( a, b, ML, MD, m, n )
      */
1746  void mcholesky_sol ( double *ML, double *MD, double *R, int m, int n )
1747  {
1748      int i;
1749      //「Ly=R0」を解く
1750      if ( m==0 ) {
1751          for ( i=1; i <= n; i++ ) {
1752              R[i] = R[i] - R[i-1]*ML[i]; }
1753      //「(D(LT))R1=y」を「(LT)R1=(D-1)y=y'」に変える
1754          for ( i=1; i <= n; i++ ) {
1755              R[i] = R[i] / MD[i]; }
1756      //「 (LT)R1=y'」を解く
```

```
1757            for ( i=n-1; i >= 0; i-- ) {
1758                R[i] = R[i] - ML[i+1]*R[i+1]; }
1759        }
1760        else{
1761            for ( i=2; i <= n; i++ ) {
1762                R[i] = R[i] - R[i-1]*ML[i]; }
1763        //「(D(LT))R1=y」を「(LT)R1=(D-1)y=y'」に変える
1764            for ( i=2; i <= n; i++ ) {
1765                R[i] = R[i] / MD[i]; }
1766        //「 (LT)R1=y'」を解く
1767            for ( i=n-1; i >= 1; i-- ) {
1768                R[i] = R[i] - ML[i+1]*R[i+1]; }
1769        }
1770
1771    }
1772
1773    /*A.6. 逆べき乗法の初期ベクトル計算 R0 ( MD, R, m, n )*/
1774    void R0 ( double *MD, double *R, int m, int n )
1775    /* 対角行列Dの成分が最大となる要素だけ1であるようなベクトルを選定*/
1776    {
1777        int i, j = 1;
1778        if ( m == 0 ) {
1779            for ( i = 0; i <= n-1; i++ ) {
1780                if ( fabs(MD[i+1]) < fabs(MD[j]) ) { j = i + 1; } }
1781            }
1782        else {
1783            for ( i = 1; i <= n-1; i++ ) {
1784                if ( fabs(MD[i+1]) < fabs(MD[j]) ) { j = i + 1; } }
1785        }
1786        //R0の初期値の代入
1787        for ( i = 0; i <= n; i++ ) {
1788                if ( i == j ) { R[i] = 1.0; }
1789                else { R[i] = 0.0; }
1790            }
1791    }
1792
1793    /*A.7. 逆べき乗法の解ベクトル規格化 R_norm ( R, n )*/
1794    void R_norm ( double *R, int n )
1795    {
1796        int i;
1797        double s = 0;
1798        //  行列要素の２乗和
1799        for ( i = 0; i <= n; i++ )  {   s = s + R[i]*R[i];  }
1800        if ( s  != 0 )
1801        {   for ( i = 0; i <= n; i++ )  {   R[i] = R[i] / sqrt(s);  }   }
1802    }
1803
1804    /*A.8. 固有値計算 (Rayleigh quotient) Eigen ( R, a, b, m, n )*/
1805    double Eigen ( double *R, double *a, double *b, int n, int m )
1806    {
1807        // Rベクトルを規格化しているため内積は1
1808        int i;
1809        double s=0;
1810        if ( m == 0 ) {
1811            s = ( R[0]*b[0] + R[1]*a[1] )*R[0];
1812            for ( i = 1; i < n; i++ ) {
1813                s += ( R[i-1]*a[i] + R[i]*b[i] + R[i+1]*a[i+1] )*R[i];  }
1814            s += ( R[n-1]*a[n] + R[n]*b[n] )*R[n];
1815            return s;
1816        }
1817        else {
1818            s = ( R[1]*b[1] + R[2]*a[2] )*R[1];
1819            for ( i = 2; i < n; i++ ) {
1820                s += ( R[i-1]*a[i] + R[i]*b[i] + R[i+1]*a[i+1] )*R[i];  }
1821            s += ( R[n-1]*a[n] + R[n]*b[n] )*R[n];
1822            return s;
1823        }
1824    /*
```

```
1825        s = b[0]*R[0]*R[0];
1826        for ( i = 1; i <= n; i++ ) {
1827            s = s + (  b[i]*R[i]*R[i] + 2.0*a[i]*R[i-1]*R[i] ); }
1828    */
1829    }
1830
1831    /*A.9. 群遅延計算用関数 dbdk_bunbo ( R, D, w, m, n )*/
1832    /* 入力パラメータ（横方向電場分布，コア径，伝搬定数，要素分割数）に対するm次モード群遅延計算式の分母分子を計算する */
1833
1834    // 横方向電場成分 R[0]~R[n]，規格化伝搬定数 w，規格化コア径 D，方位角モード次数 m，分割数 n
1835    double dbdk_bunbo ( double *R, double D, double w, int m, int n )
1836    {
1837        int i;
1838        double s=0;
1839        for ( i = 0; i <= n-1; i++ )
1840        { s = s + (1.0/12.0) * ((D*D)/(n*n)) * ( (double)(4*i+1)*R[i]*R[i] + 2.0*(double)(2*i+1)*R[i]*R[i+1] + (double)(4*i+3)*R[i+1]*R[i+1] ); }
1841        if ( m == 0) {
1842            return s + (( bessk(1,w*D)*bessk(1,w*D) / (bessk(0,w*D)*bessk(0,w*D))) - 1.0) * ((D*D)*(R[n]*R[n]) / 2.0); }
1843        else {
1844            return s + (( bessk(m-1,w*D)*bessk(m+1,w*D) / (bessk(m,w*D)*bessk(m,w*D))) - 1.0) * ((D*D)*(R[n]*R[n]) / 2.0); }
1845    }
1846
1847    /*A.10. 群遅延計算用関数     dbdk_bunshi ( R,q2, D, w, m, n )*/
1848    /* 入力パラメータ（横方向電場分布，コア径，伝搬定数，要素分割数）に対するm次モード群遅延計算式の分母分子を計算する */
1849
1850    // 横方向電場成分 R[0]~R[n]，規格化伝搬定数 w，規格化コア径 D，方位角モード次数 m，分割数 n
1851    // 屈折率分散パラメータ q2[0]~q2[n] ( = n*(d(kn)/dk) )
1852    double dbdk_bunshi ( double *R, double *q2, double D, double w, int m, int n)
1853    {
1854        int i;
1855        double s=0;
1856        for ( i=0; i<=n-1; i++ )
1857        { s = s + (1.0/12.0) * ((D*D)/(n*n))* ( ((double)(3*i)+3.0/5.0)*q2[i]*R[i]*R[i] + ((double)i+2.0/5.0)*(2.0*q2[i]*R[i+1]+q2[i+1]*R[i])*R[i] + ((double)i+3.0/5.0)*(q2[i]*R[i+1]+2.0*q2[i+1]*R[i])*R[i+1] + ((double)(3*i)+12.0/5.0)*q2[i+1]*R[i+1]*R[i+1]); }
1858        if ( m == 0) {
1859            return s + q2[n] * (( bessk(1,w*D)*bessk(1,w*D) / (bessk(0,w*D)*bessk(0,w*D))) - 1.0) * ((D*D)*(R[n]*R[n]) / 2.0); }
1860        else {
1861            return s + q2[n] * (( bessk(m-1,w*D)*bessk(m+1,w*D) / (bessk(m,w*D)*bessk(m,w*D))) - 1.0) * ((D*D)*(R[n]*R[n]) / 2.0); }
1862    }
1863
1864    /* A.13. 整数ベクトル領域確保用関数 dvector (i, j) */
1865    int *dintvector ( int i, int j ) {
1866        int *a;
1867        if ( ( a = (int *) malloc ( (j -i+1)*sizeof (int) ) ) == NULL )
1868            { printf ("Memory cannot be allocated !¥n"); exit (1); }
1869        return (a-i); }
1870
1871    /* A.14. 整数ベクトル領域解放用関数 free_dvector (a, i) */
1872    void free_dintvector ( int *a,  int i ) {
1873        free ( (void*) (a+i) ); }
1874
1875    /* A.15. 実数ベクトル領域確保用関数 dvector (i, j) */
1876    double *drealvector ( int i, int j ) {
1877        double *a;
1878        if ( ( a = (double *) malloc ( (j -i+1)*sizeof (double) ) ) == NULL )
1879            { printf ("Memory cannot be allocated !¥n"); exit (1); }
```

```
1880      return (a-i); }
1881
1882  /* A.16. 実数ベクトル領域解放用関数 free_dvector (a, i) */
1883  void free_drealvector ( double *a,  int i ) {
1884      free ( (void*) (a+i) ); }
1885
1886  /* A.17. 実数行列領域確保用関数 dmatrix ( nr1, nr2, nl1, nl2 ) */
1887  double **dmatrix ( int nr1, int nr2, int nl1, int nl2 ) {
1888      // nrow: 行の数, ncol: 列の数
1889      double **a;
1890      int i, nrow, ncol;
1891      nrow = nr2 - nr1 +1;
1892      ncol  = nl2 - nl1 +1;
1893      /* 行の確保 */
1894      if ( ( a = (double **) malloc ( nrow*sizeof (double*) ) ) == NULL )
1895          { printf ("Memory cannot be allocated !¥n"); exit (1); }
1896      a = a - nr1; // 行をずらす
1897      /* 列の確保 */
1898      for ( i = nr1; i <= nr2; i++ ) a[i] = (double *) malloc (ncol*sizeof (double) );
1899      for ( i = nr1; i <= nr2; i++ ) a[i] = a[i] - nl1;   // 列をずらす
1900      return (a); }
1901
1902  /* A.18.実数行列領域解放用関数 free_dmatrix ( a, nr1, nr2, nl1, nl2 ) */
1903  void free_dmatrix ( double **a, int nr1, int nr2, int nl1, int nl2 ) {
1904      int i;
1905      for ( i = nr1; i <= nr2; i++ ) free ( (void*) (a[i] + nl1) );
1906      free ( (void*) (a+nr1) );
1907  }
1908
1909  /* A.19. 整数ベクトル初期化関数 init_vector ( a, nr1, nr2 ) */
1910  void init_intvector ( int *a, int nr1, int nr2 ) {
1911      int i;
1912      for ( i = nr1; i <= nr2; i++ ) {    a[i] = 0; }
1913  }
1914
1915  /* A.20. 整数ベクトル初期化関数 init_vector ( a, nr1, nr2 ) */
1916  void init_realvector ( double *a, int nr1, int nr2 ) {
1917      int i;
1918      for ( i = nr1; i <= nr2; i++ ) { a[i] = 0.0; }
1919  }
1920
1921  /* A.21. 実数行列初期化関数 init_vector ( a, nr1, nr2, nl1, nl2 ) */
1922  void init_realmatrix (double **a, int nr1, int nr2, int nl1, int nl2 ) {
1923      for ( int i = nr1; i <= nr2; i++ ) {
1924          for ( int j = nl1; j <= nl2; j++ ) { a[i][j] = 0.0; } }
1925  }
1926
1927  /* 畳み込み積分 convolution (n1, P1, n2, P2) */
1928  double* convolution ( int n1, double* P1, int n2, double* P2 ) {
1929      int i, j;
1930      double* R;
1931  //  R = (double*) malloc ( sizeof (double) *(n1+n2+1));
1932      if ( ( R = (double *) malloc ( (n1+n2+1)*sizeof (double) ) ) == NULL )
1933      { printf ("Memory cannot be allocated !¥n"); exit (1); }
1934      for ( i=0; i<=n1+n2; i++ ) { R[i] = 0.0; }
1935      for ( i=0; i<n1; i++ ) {
1936          for( j=0; j<=n2; j++ ) { R[i+j] = R[i+j] + P1[i]*P2[j]; }}
1937      //for( j=0; j<=n2; j++ ) { R[i+j]+=P1[i]*P2[j]; }}
1938      return R; }
1939
1940  /* ディレクトリ作成関数 */
1941  void mkdir(char dirname[]) {
1942      struct stat statBuf;
1943      if (stat(dirname, &statBuf) != 0){
1944          if (_mkdir(dirname) != 0) {
1945              printf("ディレクトリ %s の作成に失敗しました。¥n", dirname);
1946              system("pause");    }}}
```

```
1947
1948 //! 「既存ディレクトリ削除後、ディレクトリ作成関数」失敗③
1949 /*  「既存ディレクトリ削除後、ディレクトリ作成関数 失敗①」作成後、「〃失敗②」を以下のように作成したが、
1950     ディレクトリ内ファイル削除の失敗を結局解決できず断念
1951 void delmkdirconfirm(char dirname[]) {
1952     struct stat statBuf;
1953     if (stat(dirname, &statBuf) == 0) {
1954         int i;
1955         char* str;
1956         sprintf(str, "rd /s %s", dirname);  //! sprintf(str, "rd /s /q %s", dirname);
1957         system(str);
1958
1959         WIN32_FIND_DATA findData;
1960         HANDLE hFind;
1961     }
1962     if (_mkdir(dirname) != 0) {
1963         printf("ディレクトリ %s の作成に失敗しました。¥n", dirname);
1964         system("pause");    }
1965 }*/
1966
1967 //! 「既存ディレクトリ削除後、ディレクトリ作成関数」失敗②
1968 /*  「既存ディレクトリ削除後、ディレクトリ作成関数 失敗①」作成後、「〃失敗②」を以下のように作成したが、
1969     ディレクトリ内ファイル削除の失敗を結局解決できず断念
1970 void delmkdir(char dirname[]) {
1971     struct stat statBuf;
1972     if (stat(dirname, &statBuf) == 0) {
1973         int count=0;        char listfilename[128][64];
1974         TCHAR buf[256];
1975         char str[256];
1976         sprintf(str, "%s¥*.csv", dirname);
1977         printf("%s¥n", str);
1978 #ifdef UNICODE
1979         MultiByteToWideChar(CP_OEMCP, MB_PRECOMPOSED, str, strlen(str), buf, (sizeof buf) / 2);
1980         //もしくはmbstowcs(buf,str,(sizeof buf)/2);
1981         printf("%s", buf);
1982 #else
1983         strcpy(buf, str);
1984 #endif
1985         WIN32_FIND_DATA findData;
1986         HANDLE hFind;
1987         hFind = FindFirstFile(buf, &findData);
1988         if (hFind != INVALID_HANDLE_VALUE) {
1989             do {
1990                 // ここにファイルごとの処理を記載
1991                 count += 1;
1992                 printf("%d¥n", count);
1993                 //_tprintf("%s¥n", findData.cFileName); //_tprintfとprintfは同一
1994                 sprintf(listfilename[count], "%s¥%s", dirname, findData.cFileName);
1995
1996                 printf("%s¥n", listfilename[count]);
1997             } while (FindNextFile(hFind, &findData));
1998             FindClose(hFind);   }
1999         for (int cnt = 1; cnt <= count; cnt++) {
2000                 _rmdir(listfilename[cnt]);  }
2001         _rmdir(dirname);
2002     }
2003     if (_mkdir(dirname) != 0) {
2004         printf("ディレクトリ %s の作成に失敗しました。¥n", dirname);
2005         system("pause");    }
2006 }*/
2007
2008 //! 「既存ディレクトリ削除後、ディレクトリ作成関数 失敗①」
2009 //  「既存ディレクトリ削除後、ディレクトリ作成関数」 を以下のように作成したが、ディレ
```

```
        クトリ内のファイル削除に失敗するため変更が必要
2010  /*void delmkdir(char dirname[]) {
2011      struct stat statBuf;
2012      if (stat(dirname, &statBuf) == 0) {
2013          _rmdir(dirname);    }
2014      if (_mkdir(dirname) != 0) {
2015          printf("ディレクトリ %s の作成に失敗しました。¥n", dirname);
2016          system("pause");    }}*/
```