電磁界最適化ソフトウェア
取扱説明書

株式会社 EEM

2012 年 3 月

目次

1. セットアップ

1.1 ソフトウェア概要

本ソフトウェアは電磁界の最適化を行うものです。
最適化手法には滑降シンプレックス法[1]を用います。
電磁界の計算部は差分法による電磁界シミュレータ EEM-FDM を用います。

1.2 インストール

(1)EEM-FDM
電磁界シミュレータ EEM-FDM を付属の説明書を参考にインストールして下さい。

(2)最適化ソフトウェア
適当な作業フォルダを作成し(EEM-FDM と別の場所でもかまいません)、本最適化ソフトウェア CD の全ファイルをそのままコピーして下さい。

(3)コンパイラ
Microsoft Visual Studio をインストールします。
本説明書ではコマンドラインでの作業について説明します。統合開発環境で作業する場合は適当に読み替えて下さい。
スタートメニューの[Microsoft Visual Studio 2010][Visual Studio Tools][Visual Studio コマンドプロンプト(2010)]を起動します。
コマンドラインで
> cl
と入力してロゴが出ることを確認して下さい。(以下、先頭の>はコマンドプロンプトを表します)

(4)コマンドプロンプト
コマンドプロンプトウィンドウのサイズ、色、フォントを変えるには、[Visual Studio コマンドプロンプト(2010)]を右クロック＋[管理者として実行]として起動し、左上のアイコンをクリックし[プロパティ]で変更します。
また[Visual Studio コマンドプロンプト(2010)]ショートカットの[プロパティ]の[作業フォルダ]に(2)で作成したフォルダを入力しておくと、初めに cd コマンドでフォルダを移動する手間が省けます。
コマンド履歴はカーソルキー↑↓で呼び出すことができます。
またコマンドライン入力中に Tab キーでファイル名を補完することができます。

(5)エディタ
ソースコードの編集にはテキストエディタを使用します。Windows 付属のメモ帳でもかまいませんが、高機能なエディタを用いると作業の能率が上がります。

[1]W.H.Press 他(丹慶勝市他訳)"Numerical Recipes in C(日本語版)"、技術評論社、1993、第 10 章

2.　操作手順

2.1　最適化作業の準備

(1)EEM-FDM での検討
最適化するモデルのプロトタイプを EEM-FDM で作成し、パラメータを変えて計算し、計算対象の特性を十分把握しておきます。
計算時間を短縮するために、必要最低限のメッシュはどの程度か、必要最低限のタイムステップ数はどの程度か調査します。その他の計算上のパラメータについても確定します。アンテナの計算の計算時間を短縮するには仮想抵抗が有効な場合があります。
最適化作業を効率よく行い、よりよい結果を得るにはこの作業が最も重要です。

(2)最適化するパラメータの抽出
所望の特性を得るために最適化すべきパラメータを決定します。なるべく少ないパラメータで、かつ目的を達成できるようなパラメータを選択することが重要です。
また、パラメータの最適値がどの範囲の中にあるか調べておきます。
最適計算を行った後、結果が思わしくないときは本ステップに戻ります。

2.2　計算モデルの編集

最適化するパラメータを EEM-FDM の計算部に渡すために、ソースコード setdat.c を編集します。
> notepad setdat.c　　(以下、エディタがメモ帳以外のときは notepad を適当に変えて下さい)
詳しくは 3 章で説明します。

2.3　評価関数の編集

最適化の対象となる評価関数を編集します。ソースコード getval.c を編集します。
> notepad getval.c
複数の特性を同時に最適化するには、それらの重み付けには試行錯誤が必要になります。
詳しくは 4 章で説明します。

2.4　コンパイル・リンク

コマンドラインで
> nmake
と行います。
実行プロプラム ds.exe ができます。
プログラム nmake.exe は更新したファイルのみをコンパイルし、その後リンクします。コンパイルのルールを記述したファイルが Makefile です。

2.5　最適化制御

最適化作業を制御するためのファイル opt.dat を編集します。
> notepad opt.dat

opt.dat の意味は以下の通りです。数値の後の文字列はコメントです。

表 1　最適化制御ファイル opt.dat

```
30      max iterations          最大計算回数(1)
50      no. of initial set      初期シンプレックスを決めるための計算回数(2)
1       seed (>0)               乱数の種(正の整数)(3)
0       debug (0/1/2)           デバッグ用フラグ(4)
2       no. of threads          計算のスレッド数(5)
1e-3    convergence             最適計算の収束判定条件(6)
3       no. of parameters       最適するパラメータの数(7)
30 100                  最適するパラメータの下限と上限(以下パラメータの数だけ続きます)(8)
30 100
30 100
```

(1)最大計算回数
最適化パラメータの数に応じて数十～数百の値を代入します。

(2)初期計算回数
最適化パラメータの数に応じて数十～数百の値を代入します。

(3)乱数の種
初期シンプレックスを探索するための乱数を決めます。乱数は線形合同法[1]で作られる擬似乱数でその周期は 1664501 です。乱数の種が 1 違うだけでまったく違う乱数が生成されます。乱数の種が同じであれば同じ乱数を生成しますので、同じ初期シンプレックスができます。その後の最適計算は乱数を用いない決定論的なものですから、結局乱数の種が同じであれば同じ結果が得られます。

(4)デバッグ用フラグ
2 を代入すると、EEM-FDM 用データ opt.out を作成し、計算せずに終了します。これは 2.2 の setdat.c が正しく編集されているかどうかの確認に使用します。EEM-FDM で opt.out を開きデータを確認します。
1 を代入すると、EEM-FDM 用データ opt.out を一つ作成して計算します。これは 2.3 の getval.c が正しく編集されているかどうかの確認に使用します。画面に表示される評価関数値を確認します。
0 を代入すると、最適計算の本計算を開始します。

(5)計算のスレッド数
計算時に使用するスレッド数です。CPU の物理的コア数以下の数字を入力して下さい。この数字が大きいほど計算時間が短縮されます。ただし、問題のサイズが小さいときは計算時間の短縮度は小さくなります。

(6)収束判定条件

シンプレックスのすべての頂点の評価関数の差がこの値以下になると計算を終了します。計算を打ち切るには、(1)の最大計算回数またはこの収束判定条件を使います。

(7)最適パラメータの数
2.1で決めたパラメータの数を代入します。

(8)パラメータの下限と上限
各パラメータの最適値があると予想される下限と上限を代入します。
単位は何でも構いませんが、setdat.cでEEM-FDMが認識するMKSA単位に変換する必要があります。
初期計算に用いられるパラメータの値は、下限と上限の間でランダムにとられます。

2.6 計算

コマンドラインで
> ds
と行います。
最適化計算が開始し、画面に計算経過が表示されます。計算経過の意味は以下の通りです。

表2 計算経過の出力

```
No. of parameters  = 3          最適化するパラメータの数
No. of initial set = 50         初期計算回数
Max iterations     = 30         最大計算回数
Seed of eandomness = 1          乱数の種
No. of threads     = 2          計算のスレッド数
Tolerance          = 0.001      収束判定条件
    1*  0.0774 = +0.0701  +0.0073       計算回数、評価関数値、評価関数の各項
    2*  0.0338 = +0.0291  +0.0047
    3*  0.0268 = +0.0184  +0.0084
    4*  0.0159 = +0.0140  +0.0020
    5   0.0295 = +0.0225  +0.0071
 (略)
   47   0.0202 = +0.0120  +0.0082
   48   0.0639 = +0.0583  +0.0057
   49   0.0217 = +0.0163  +0.0054
   50   0.0554 = +0.0470  +0.0084
=== initial simplex ===                 初期シンプレックスが得られた
    1   0.0025          以下、初期シンプレックスの各頂点の評価関数値(パラメータの数+1個)
    2   0.0043
    3   0.0045
    4   0.0077
=== optimization start ===              最適化計算の開始
   51   0.0115 = +0.0105  +0.0010       計算回数(累計)、評価関数値、評価関数の各項
   52   0.0029 = +0.0025  +0.0003
   53   0.0304 = +0.0295  +0.0009
```

```
   54    0.0234 = +0.0230   +0.0004
(略)
   77    0.0007 = +0.0003   +0.0004
   78*   0.0007 = +0.0004   +0.0003
   79    0.0007 = +0.0005   +0.0003
   80    0.0007 = +0.0002   +0.0005
```

計算回数の後ろの*印は評価関数の最適値が更新されたことを表します。このとき対応するEEM-FDMデータがopt.outファイルに、最適パラメータ値がopt.parファイルに出力されます。
最適計算の途中で別途EEM-FDMを起動してopt.outを開いて計算し、最適化が正しく行われているかどうか確認することができます。
最適パラメータ出力ファイルopt.parは以下の通りです。一行に最適パラメータ値が書かれています。

表3　最適パラメータ出力ファイルopt.par

```
84.475328 65.952406 30.248900
```

3.　計算モデルの編集

最適すべきパラメータをEEM-FDMの計算部に渡すためのファイルsetdat.cの一例を表4に示します。
各関数の仕様は「EEM-FDMデータ作成ライブラリ(C版)　取扱説明書」を参考にして下さい。
このうち、最初に初期化する"fdm_init"と最後にファイル出力する"fdm_outdat"関数は不要です。
必須項目はメッシュ、形状データ、波源(給電点または平面波入射)、周波数です。その他必要なものを入力して下さい。呼び出さないときは既定値が設定されるものもあります。
各ブロックの順序は任意ですが、表4の順序を推奨します。
setdat.cファイルは最適化するモデルごとに作成し、適当な名前を付けてsetdatフォルダに保存することを推奨します。

下記の例は、中央に3辺の長さをパラメータとする一つの直方体を置き、＋X方向(θ=90度、φ=0度)から垂直偏波の平面波が入射したモデルです。

表4　setdat.cの一例

```
#include "fdm_datalib.h"                        関数のプロトタイプ宣言(必須)

void setdat(int nparm, const double *p) 変更不要、nparm:パラメータ数、p[0],p[1],...:パラメータ
{
        double x1, x2, y1, y2, z1, z2;
        double lx, ly, lz;
        const double x0 = -150e-3;
        const double x3 = +150e-3;
        const double y0 = -150e-3;
        const double y3 = +150e-3;
        const double z0 = -150e-3;
        const double z3 = +150e-3;
        const double d = 10e-3;

        lx = p[0] * 1e-3;  // mm -> m
        ly = p[1] * 1e-3;  // mm -> m
        lz = p[2] * 1e-3;  // mm -> m

        fdm_title("DS-EEM-FDM");                        タイトル(オプション)

        fdm_domain(1);   // TIME & SINE                 計算方法

        // mesh                                         メッシュ

        x1 = -lx / 2;
        x2 = +lx / 2;
        fdm_xsection(4, x0, x1, x2, x3);                X方向
        fdm_xdivision(3, (int)((x1-x0)/d+0.5), (int)((x2-x1)/d+0.5), (int)((x3-x2)/d+0.5));
```

```
    y1 = -ly / 2;
    y2 = +ly / 2;
    fdm_ysection(4, y0, y1, y2, y3);                Y 方向
    fdm_ydivision(3, (int)((y1-y0)/d+0.5), (int)((y2-y1)/d+0.5), (int)((y3-y2)/d+0.5));

    z1 = -lz / 2;
    z2 = +lz / 2;
    fdm_zsection(4, z0, z1, z2, z3);                Z 方向
    fdm_zdivision(3, (int)((z1-z0)/d+0.5), (int)((z2-z1)/d+0.5), (int)((z3-z2)/d+0.5));

    // material                                     物性値(誘電体・磁性体を使用するとき必要)

    // geometry                                     形状データ

    fdm_unit6(1, 1, x1, y1, z1, x2, y2, z2);        直方体

    // incidence                                    波源

    fdm_incidence(90, 0, 0);                        平面波入射

    // frequency                                    周波数

    fdm_freq(3e9, 3e9, 0);                          近傍界・遠方界用周波数

    // solver                                       計算制御

    fdm_iteration(0.001, 500, 0);                   収束判定条件他
}
```

4.　評価関数の編集

モデルの特性を定量的に評価するためのファイル getval.c の一例を表 5 に示します。
2 つの出力引数 nfnc と wfnc に値を代入します。
nfnc は評価関数の項目の数、wfnc は評価関数の各項目の値(重みつき)です。wfnc の和が最適化の判定に使用される評価関数値です。
本ソフトウェアでの「最適」とは評価関数が最小であることを意味しますので、大きい方が望ましい物理量については、符号を反転する、逆数をとるなどの操作を行って下さい。
getval.c ファイルは複数のモデルで共通することがありますので、一つのファイルで複数のモデルに対応することを推奨します。

下記の例は＋X 方向(θ=90 度, φ=0 度)と+Y 方向(θ=90 度, φ=90 度)の遠方界の和を最小にするものです。

表 5 評価関数 getval.c

```
#include <stdio.h>
#include <math.h>
#include "complex.h"            関数のプロトタイプ宣言(必須)
#include "valfs.h"              関数のプロトタイプ宣言(必須)

void getval(int *nfnc, double wfnc[])
{
        int db       = 0;
        int ifreq    = 0;
        int comp     = 0;
        double theta = 90.0;
        wfnc[0] = + 1.0 * em_gain(db, ifreq, comp, theta,  0.0);
        wfnc[1] = + 1.0 * em_gain(db, ifreq, comp, theta, 90.0);
        *nfnc = 2;
}
```

getval.c の各関数の詳細は表 6 の通りです。
関数名はすべて"em_"で始まります。
実数の引数と関数値はすべて倍精度(double)で単位は MKSA です。
構造体"z_complex_t"は倍精度複素数を表し、メンバー".r",".i"が実部と虚部を表します.
複素数関係の関数の使い方は complex.h のプロトタイプ宣言を参考にして下さい。

表 6　getval.c 用の関数

(1)入力インピーダンス[Ω]

```
z_complex_t em_zin(int ifeed,           給電点番号 0,1,2...
                   int ifreq);          周波数番号 0,1,2...
```

(2)反射係数 Γ

```
double em_reflection(int db,            0/1:線形/dB  dB=20*log10(線形)
                     int ifeed,         給電点番号 0,1,2...
                     int ifreq,         周波数番号0,1,2...
                     double z0);        給電線の特性インピーダンス[Ω]
```

(3)VSWR

```
double em_vswr(int ifeed,               給電点番号 0,1,2...
               int ifreq,               周波数番号0,1,2...
               double z0);              給電線の特性インピーダンス[Ω]
```

(4)利得または散乱断面積

```
double em_gain(int db,                  0/1:線形/dB  dB=10*log10(線形)
               int ifreq,               周波数番号0,1,2...
               int comp,                成分  0:絶対値、1:θ 成分、2:φ 成分
               double theta,            θ[度]
               double phi);             φ[度]
```

(5)軸比

```
double em_axr(int db,                   0/1:線形/dB  dB=20*log10(線形)
              int ifreq,                周波数番号0,1,2...
              double theta,             θ[度]
              double phi);              φ[度]
```

(6)ビーム幅[度]

```
double em_beamwidth(int ifreq,          周波数番号0,1,2...
                    int comp,           成分  0:絶対値、1:θ 成分、2:φ 成分
                    int ndiv,           観測面の分割数
                    int plane,          観測面の種類  1:θ 一定面、2:φ 一定面
                    double angle);      一定の θ または φ[度]
```

(7)サイドローブレベル

```
double em_sidelobe(int db,              0/1:線形/dB  dB=10*log10(線形)
                   int ifreq,           周波数番号0,1,2...
                   int comp,            成分  0:絶対値、1:θ 成分、2:φ 成分
                   int ndiv,            観測面の分割数
                   int plane,           観測面の種類  1:θ 一定面、2:φ 一定面
                   double angle);       一定の θ または φ[度]
```

(8)前後比

```
double em_fbr(int db,                   0/1:線形/dB  dB=10*log10(線形)
              int ifreq,                周波数番号0,1,2...
              int comp,          成分  0:絶対値、1:θ 成分、2:φ 成分
              double theta,             前方の θ[度]
              double phi);              前方の φ[度]
```

(9)結合度

```
double em_couple(int db,                  0/1:線形/dB  dB=10*log10(線形)
                 int iload,               集中定数(=抵抗)番号0,1,2...
                 int ifreq);              周波数番号0,1,2...
```

(10)Sパラメータ

```
z_complex_t em_spara(int iport,           観測点番号0,1,2...(順にS11,S21,S31,...)
                     int ifreq;           周波数番号0,1,2...
```

(11)観測点の電界強度[V/m]

```
z_complex_t em_point(int id,              向き  0:X方向 1:Y方向 2:Z方向
                     int i,               X座標の節点番号
                     int j,               Y座標の節点番号
                     int k,               Z座標の節点番号
                     int ifreq);          周波数番号0,1,2...
```

5.　EEM-FDM での評価

以上で得られる最適されたデータファイル opt.out を EEM-FDM で開いて各種特性の詳細を確認することができます。
図 1 は EEM-FDM でデータを開いた状態、図 2 は Z 面の遠方界パターンです。
図 2 より+X 方向と+Y 方向の遠方界が小さくなっていることがわかります。

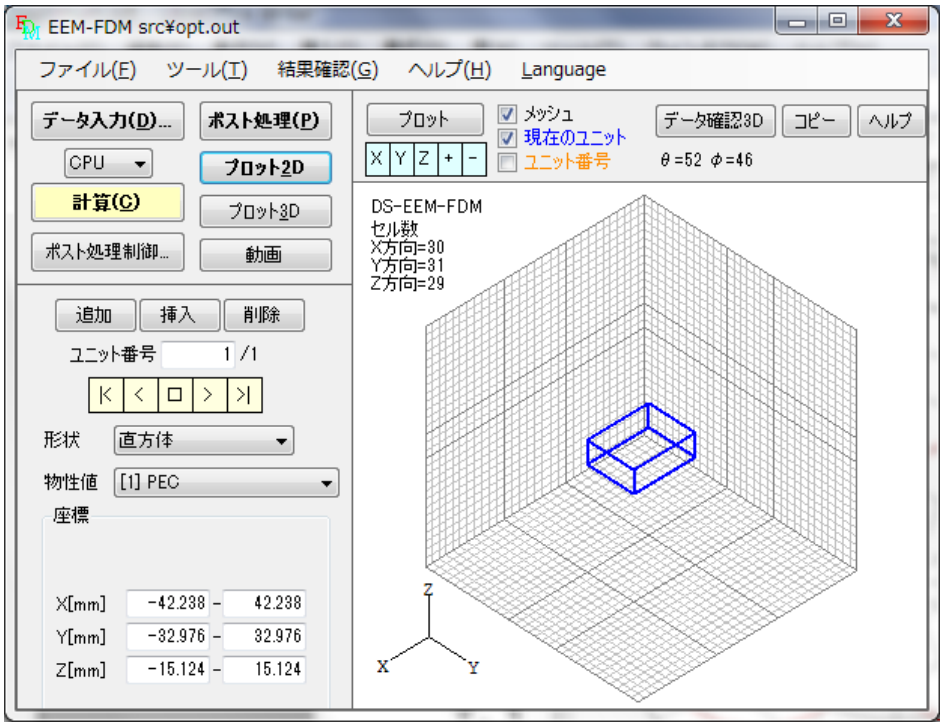

図 1 最適化されたデータ opt.out を EEM-FDM で開いた状態

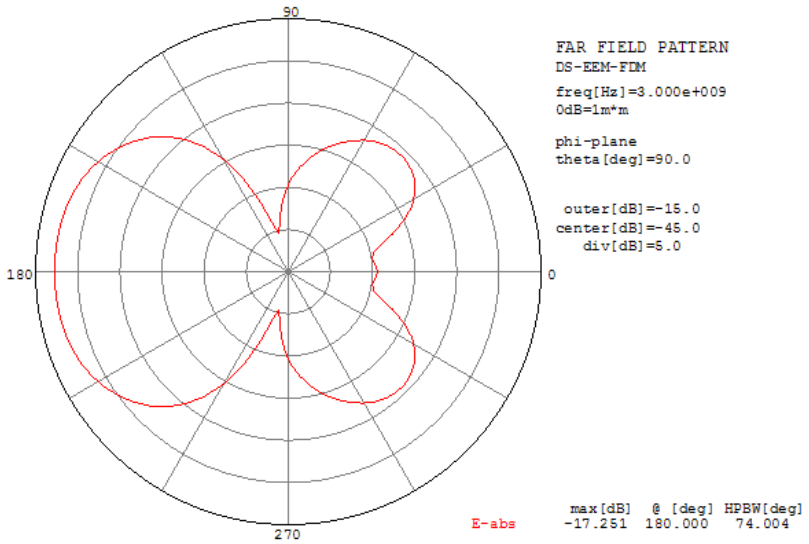

図 2　遠方界パターン(Z 面)

6. ファイル構成

本ソフトウェアのファイル構成は以下の通りです。
*印のついたものがユーザーが編集するファイルです。

表7 ファイル構成

Makefile	Makefile(コンパイル･リンク管理ファイル、nmake.exe と連動)
ds.exe	最適計算実行プログラム
opt.dat	*最適計算制御ファイル
opt.out	最適化された EEM-FDM 用ファイル(出力)
opt.par	最適化されたパラメータ(出力)
setdat.c	*最適パラメータを EEM-FDM に渡すソースコード
getval.c	*評価関数を計算するソースコード
optimum.pdf	本取扱説明書(PDF)
fdm_datalib_c.odf	EEM-FDM データ作成ライブラリ(C 版)取扱説明書(PDF)
optdat/	opt.dat を名前を変えて保存するところ
setdat/	setdat.c を名前を変えて保存するところ
(以下のソースコードは変更しないで下さい)	
Main.c	主プログラム
amoeba.c	最適化を行う
funk.c	評価関数を計算する
iniset.c	初期シンプレックスを求める
optdat.c	opt.dat を読み込む
valfs.h	getval.c 用関数プロトタイプ宣言
valfs.c	getval.c 用関数(汎用)
valfs_fdm.c	getval.c 用関数(EEM-FDM 専用)
complex.h	複素数関数プロトタイプ宣言
fdm_datalib.h	EEM-FDM データ作成ライブラリ(setdat.c 作成用)
fdm_datalib.c	EEM-FDM データ作成ライブラリ(setdat.c 作成用)
fdm.h	EEM-FDM ヘッダファイル
fdm2.obj	EEM-FDM 計算部(オブジェクトファイル)