超広帯域高周波トランスの伝送特性解析

平成19年度

三重大学大学院工学研究科
博士前期課程 物理工学専攻

加 藤 弘 晃

修士論文

超広帯域高周波トランスの伝送特性解析

平成19年度

三重大学大学院 工学研究科　物理工学専攻

加藤弘晃

目 次

第1章　序論

社会のIT化・ユビキタス化に伴い通信技術の革新はとどまることを知らない. 情報通信の高度化に伴って, 一度に大量の情報をしかも高速に処理しなければならない.CATVなど周波数多重を用いるシステムでは, そうした通信の大容量化に対応するために, 利用する周波数帯域が高域側にシフトしつつ広帯域化することが求められている. 通信システムの高域化と広帯域化は, つまりは通信デバイスのそれを意味しているが, 高周波であるがゆえにその設計を複雑かつ困難なものにしている.

CATVシステムにおける重要なデバイスの一つに高周波分配器がある. 分配器は, 一つの入力を複数の出力に分けることを役割とするデバイスである. またCATVネットワークの中で最も広く使われている受動素子であり, 強磁性体であるフェライトを用いた高周波トランスから構成されている. 現在,CATVシステムは多チャンネルの情報を伝送するため, そこで利用される高周波分配器は10MHzから200MHzあるいはそれ以上といった超広帯域での動作が必要であり, それに対応したデバイス設計が求められる.

こうした高周波デバイスの設計法の一つとして試作を繰り返す方法もあるが, コストや時間の面でデメリットが大きい. これに代わる手法として, コンピューターを利用した解析 (CAE:computer-aided engineering) が有効と考えられる.

電磁界数値解析の分野において, 解析ツールとして使用される手法としては, 有限要素法, 境界要素法,FDTD法などが挙げられる. 複雑な形状や構造を持った電磁波解析では, これら各種数値計算法が極めて大きな威力を発揮する. 本研究の目的は, こうした電磁界シミュレーション技術を高周波デバイス (高周波トランス) の設計に応用することによって, 従来にない広帯域な伝送機器の開発につなげることである.

具体的に取り組んだ内容は, 下記の3つである.

1 分配器用の高周波トランスの広帯域化を実現するために, フェライトコアの形状および材料特性, 巻線の巻き方, 基板への実装方法などとトランスの伝送特性との関係についてCAEによって検討した. そして広帯域化のための指針を得ることが出来た.

2 筆者らの提案している光ファイバ伝送における波長二次歪み補償システムで利用する高周波用プッシュプル (P-P) トランスの伝送特性について, 同じくCAEを用いて解析を行ない, デバイス構造と伝送特性との関係を明らかにした.

3 高周波トランスを用いたデバイスの特性解析のため, 時間変化を追える3次元FDTDのコードを作成することとなり,C言語によるコーディング方法についてまとめた.

第2章　分配器用高周波トランスの広帯域化を目指した設計

要約

本章では,CATV システムで広く利用されている RF 分配器の伝達特性を検討した結果について述べる. 分配器が要求する動作周波数の広帯域化しており,RF 変換器の伝達特性とその特性を左右する設計パラメータの関係は非常に複雑なものとなる. そこで, コンピュータを用いた電磁界シミュレーションを行なった. 次に理論的に求めた結果を裏付けるために, その計算に基づき試作品を作製した. デバイスの特性に最も影響を与える要素は, トランスを構成する巻き線の部分であることが分かった. 適切な巻き線の巻き方を選択することによって,10MHz から 2600MHz までの超広帯域で低損失を実現した結果について,2 分配,4 分配, それぞれのトランス分けて述べる.

2.1　はじめに

近年の情報技術の進歩によって, 膨大な量のデータをやり取りするため多くの通信デバイスに高速動作, そして広帯域での動作が要求される.RF 分配器もその一つである. その役割は,1 入力を複数に分けるもので,CATV ネットワークの中で最も広く使われている受動素子であり, 強磁性体であるフェライトから作られたトランスで構成されている. 現在,CATV システムを取り巻く環境も上述したような状況にあり, 周波数多重された多くのチャンネルの情報を伝達するため,RF 分配器は 20MHz から 2600MHz までと言う超広帯域で正確な動作が要求されている. 言い換えると,RF 分配器での損失は可能な限り小さくなる必要がある. それは,RF 分配器は信号を各家庭に送るケーブルの中で, 直列に多数使用されている事実に起因する. 文献 [1]-[3] では RF 分配器の動作周波数広帯域化について, 等価回路を用いて検討されている. しかしながら, デバイスに対する RF 分配器に影響を与える多様な設計パラメータの影響は, この手法では明確に出来なかった. 広帯域分配器を作製するためには, 設計パラメータの影響に関してより深く理解することが重要である. 本論文では, フェライトコアの透磁率や形, その表面などのパラメータが, 伝達特性にどのように影響を与

えるかを検討した. これらの設計パラメータと伝達特性の関係を CAE を用いて詳細に検討し, それに基づきデバイスを試作した. そしてシミュレーションと実測双方において, 上記の超広帯域において非常に精度のよい結果を得ることが出来た.

2.2　CATV システム用 RF 分配器

2.2.1　構造

まず CATV システムに使用される 2 分配器,4 分配器の動作周波数広帯域化を説明する. それらは図 2.1 に示してあるように 3 個若しくは 5 個の RF トランスで構成されている. 図 2.1(a) は 2 分配器の電気回路を表しており [1][2], それは 1 つの分配トランス (T_1) と 2 つの整合トランス (T_2) からなっている. 図 2.1(b) は 2 分配器の電気回路を表しており, それは 3 つの分配トランス (T_1, T_4) と 2 つの整合トランス (T_3)[3] からなっている (T_1,T_2,T_3,T_4 の仕様については後述する). それらのトランスは全て, ビーズ型フェライトコアに数ターンの巻き線を巻きつけたもので構成されている. 図 2.2(a) と図 2.2(b) は, 分配トランス (T_1) と整合トランス (T_3) が基板に実装された状態を模式的に描いた図である. 図中では, 巻き線の詳細を紹介するためにフェライトコアを分割してある. トランスの入出力は, マイクロストリップラインで構成されており, 巻き線はそこに半田付けされている. フェライトコアの透磁率や形や寸法, 巻き線やフェライトコアの実装方法などが分配器の伝達特性に影響する. それゆえに, デバイスの性能を評価し動作周波数を広帯域化する, 言葉を変えれば伝達の際に生じる損失を抑制するためには, これらのパラメータの寄与についての検討を行なう必要がある.

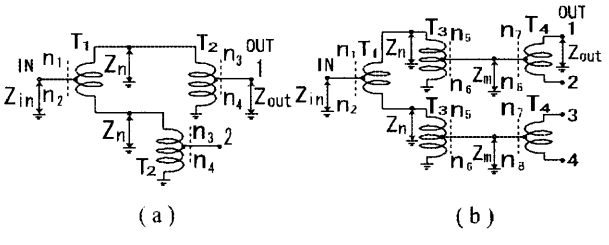

図 2.1: 等価回路 ： (a) 2 分配器 (b) 4 分配器

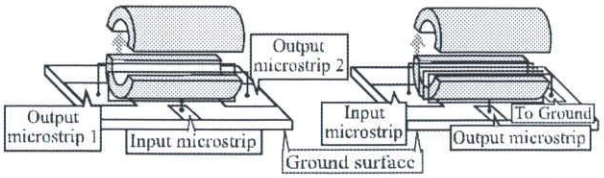

図 2.2: 構造 ： (a) 分配トランス T_1 (b) 整合トランス T_3

2.2.2　設計

RF 分配器の設計において, 実測とその結果を基にした試行錯誤は大きな意味を持っていた. 文献 [2] では,RF 分配器の伝達特性を解析するために巻き線の外部を考慮に入れた等価電気回路の活用が提案されている. しかしながら, より複雑な形状の分配器では, この手法の適応は困難であった. そのような場合, 広く構造解析に使われている CAE が強力なツールとなる. 本研究では CAE 手法を取り入れ RF 分配器の設計を行なった.

RF 分配器の基本的な特性に関するパラメータは, 伝達損失 (分配損失と過剰損失の合算である) と反射損失, そして出力間のアイソレーションである. これらの中で, 最も重要なパラメータは損失である. 分配器の過剰損失は, 分配トランスよりも整合トランスでの方が大きいことが知られている. 実際にネットワークアナライザーを用いて測定した結果を図 2.3 に示す.

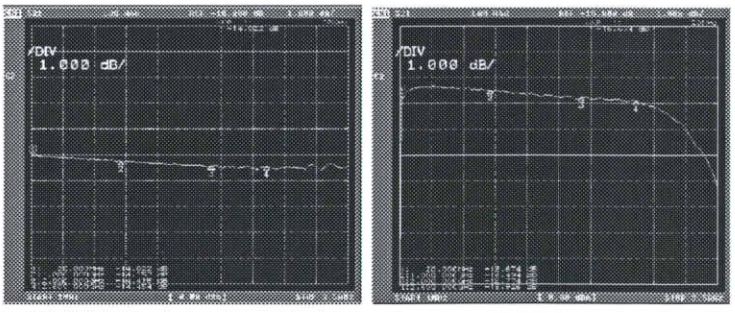

図 2.3: 過剰損失 ： (a) 分配トランス T_1 (b) 整合トランス T_3

周波数が高くなれば (GHz オーダー)[2] この傾向が顕著になることが分かる. それは, 分配トランスの巻き線中を電流がお互いに反対方向に流れ, それぞれの巻き線が作る磁場が打ち消しあうことでフェライトコア内に発生する磁束を減少させるためと考えられる. それゆえ今後の議論は, 動作

周波数帯域の広帯域化が一つの目標であるので, 整合トランス (T_3) の伝達特性を中心に進めていく.

2.3　シミュレーション

2.3.1　モデル作成

コンピュータ解析では, 電磁界シミュレーションソフトウェア JMAG-Studio(日本総研　) を利用した. このソフトウェアは有限要素法を用いて電磁界解析を行なう事も出来るツールである. 本研究では, このソフトウェアを利用し,2 分配器,4 分配器において種々のトランス伝達特性を評価する. それぞれのトランスについての設定変数は, 表 2.1 にまとめてある. またそれぞれのタイプのフェライトコアに対する磁気定数は表 2.2 に挙げたとおりである.

表 2.1: 2 分配 4 分配トランスの仕様

	T_1	T_2	T_3	T_4
Number of turns in winding turns	$n_1 = 1$ $n_2 = 1$	$n_3 = 1$ $n_4 = 3$	$n_5 = 2$ $n_6 = 2$	$n_7 = 1$ $n_8 = 1$
Input impedance	75 Ω	150 Ω	150 Ω	37.5 Ω
Output impedance	150 Ω	75 Ω	37.5 Ω	75 Ω
Core I.D.	0.8mm	1mm	1mm	0.7mm
Core O.D.	3.5mm	3.5mm	3.5mm	3.5mm
Core length	4mm	3mm	3mm	2mm
Type of core permeability	# 1	# 2 or # 3	# 2 or # 3	# 4

表 2.2: フェライトコアの透磁率分散

	#1	#2	#3	#4
μ_{r0}	2200	260	100	2000
f_r (MHz)	3.5	25	60	3.5
K	1.1	1.4	1.5	1.1

典型的な解析モデルを図 2.4 に示す. トランスの入出力はマイクロストリップラインで実現しており, そのインピーダンスは表 2.1 の値に整合するように設計されている. 計算に用いた寸法やフェ

ライトコアの磁気定数も表 2.1 に記載した. 巻き線の巻き数は, 入出力のインピーダンスに整合するように選んだ. 例えば整合トランス (T_3) では, 入出力のインピーダンスが 150 Ω と 37.5 Ω なので, 巻き線の巻き数はそれぞれ $n_5 = 2$,$n_6 = 2$ となる. このモデルは 1 次巻き線が 4 巻きであり, 二次巻き線の中心からタップして出力をとり出している. インピーダンス変換則の説明に関しては付録に詳細を記載したので, そちらを参照されたい.

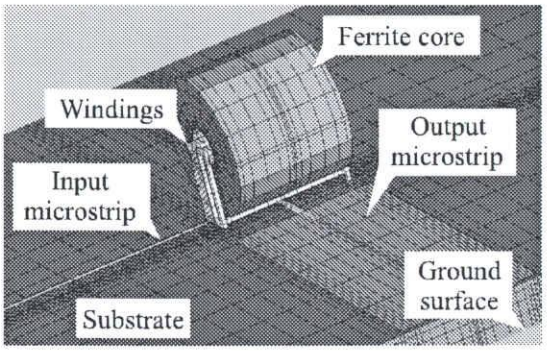

図 2.4: 整合トランスの伝達特性解析用メッシュモデル

またフェライトコアの透磁率に関しては, 市場で一般的な表 2.2 のデータを用いた. さらに透磁率の周波数依存性, すなわち分散特性に関しては式 2.1 で近似した.

$$\mu_r = \mu_r^{'} - j\mu_r^{''} = 1 + \frac{K\mu_{r0}}{1 + j\,(f/f_r)} \tag{2.1}$$

ここで定数 K は分散定数,μ_{r0} は初期透磁率,f は周波数, そして f_r 共振周波数である [4]. 具体例として, #1 タイプの透磁率分散曲線を図 2.5 に示す.

2.3.2　検討パラメータ

異なる設定変数を有するモデルを多数用意し, それらの変数が伝達特性にどのような影響を与えるか検討した. 結果の詳細は後述することとし, まずは検討パラメータをリストにまとめた. それらは次のリストにあるような 4 つの大きな分類に分けられる.

検討パラメータ

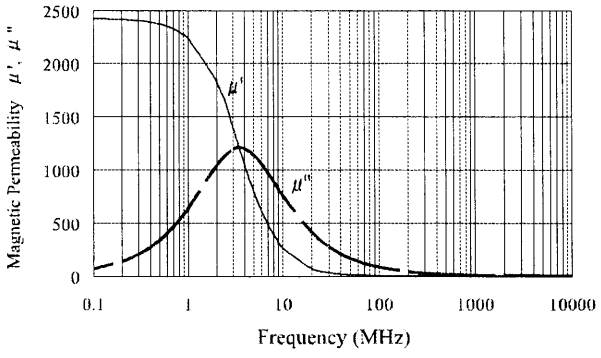

図 2.5: 計算用に設定した透磁率分散値の一例

1 フェライトコアの長さ

2 フェライトコアの巻き線の巻き方

3 フェライトコアの透磁率の変化

4 フェライトコアの偏芯

第一に, フェライトコアの内径 (I.D.), 外径 (O.D.) を 1mm,3.5mm と固定し, 径に対する長さの比 (アスペクト比) を変化させた.

第二に, 整合トランスの巻き線を変更させたモデルについても解析した. そのモデルとは, 通常の順巻きと種々の交差巻きの構成である. 交差巻きについては次章で記述する.

第三に, 使用するフェライトコアの透磁率を変更して, その変化量に応じた伝送特性の違いを検討した.

最後に, ビーズ型フェライトコアの空洞部と基板との位置関係を変化させたものを検討した. この目的は, 巻き線が基板に近づくほど, 測定値が良いと言う過去のデータを元に検討した. これに関しても詳細は次章で記する.

2.3.3　評価量 (S パラメータ) について

ここでは S パラメータについて簡単にまとめる. 一般の電気回路や電子回路では, 回路の特性を表すのに,Z パラメータ, Y パラメータ, h パラメータなどが使われる. そしてこれらのパラメータは, 電圧・電流での回路特性の測定・評価を前提としている. しかし高周波では, 低周波のように電圧や電流を測定することはほとんど不可能となる. 例えば, 電圧測定のためにプローブなどをパターンに接触させると, 回路構成が変化してしまう. そこで電圧や電流に代わる別な量での測定・評価が必要になる. 高周波領域でも安定して正確に測定可能な量は電力である. 回路に入って行く電力と, 回路から出てくる電力を関係付けたのが S パラメータである. 詳しくは図 2.6 を参照されたい.

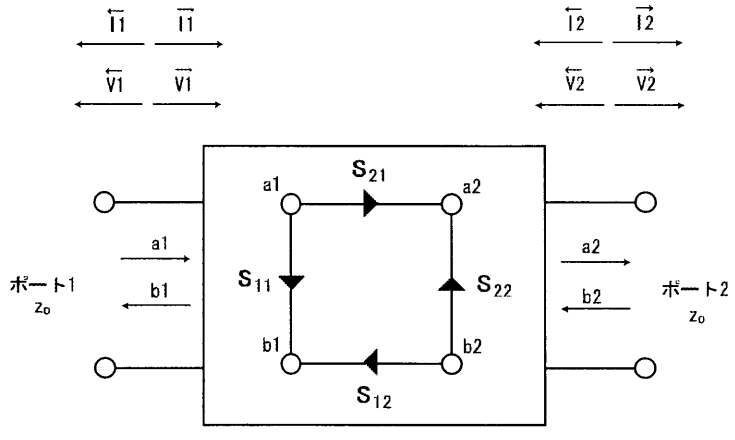

図 2.6: S パラメータ

図 2.6 に示す 2 ポート回路（2 端子対回路）で, 各ポートの電圧, 電流は, 進行波（入ってゆく波）と後退波（出てくる波）の和で表される.

$$V_n = \overleftarrow{V_n} + \overrightarrow{V_n}$$
$$I_n = \overleftarrow{I_n} + \overrightarrow{I_n}$$

次に各ポートの特性インピーダンスを Zo として,a_n,b_n を次式で定義する.

$$
\begin{aligned}
a_n &\equiv \frac{\overrightarrow{V_n}}{\sqrt{Z_0}} = \frac{\overrightarrow{I_n}}{\sqrt{Z_0}} \\
b_n &\equiv \frac{\overleftarrow{V_n}}{\sqrt{Z_0}} = \frac{\overleftarrow{I_n}}{\sqrt{Z_0}}
\end{aligned}
$$

したがって, 各ポートから流入する電力 P は,

$$P_n = |a_n|^2 - |b_n|^2$$

a_n, b_n は, 複素量で大きさと位相を持っている. 絶対値の二乗が, それぞれ流入方向, 流出方向の電力を表す. この入力波 a_n と出力波 b_n の関係を定義したものが S マトリクス（散乱行列, scattering matrix）と呼ばれている. そして S マトリクスの各要素を S パラメータと言う.

$$
\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}
$$

また各 S パラメータは以下のように定義されている.

$$
\begin{aligned}
S_{11} &= \frac{b_1}{a_1} \\
S_{21} &= \frac{b_2}{a_1} \\
S_{12} &= \frac{b_1}{a_2} \\
S_{22} &= \frac{b_2}{a_2}
\end{aligned}
$$

つまり Sii はポート i 以外を全て Zo で終端したときの, ポート i の反射係数であり,Sji はポート i 以外を全て Zo で終端したときの, ポート i からポート j への伝達係数となっている.

2.3.4　計算結果

伝達特性は前述した S パラメータ, すなわち伝送特性 S_{21} と反射特性 S_{11},S_{22} で評価される. 図 2.7 は, コアの長さ L =1,3,5mm を変化させたときの伝達特性を表している. これらの結果から, コアの長さが短くなるほど高い周波数で伝達特性が向上し, 逆に低い周波数で特性が悪化すると言える. しかし一方で, コアの長さが長くなるほど低い周波数で伝達特性が向上し, 高い周波数で伝達特性が悪化する傾向にある. 広い周波数帯域で低損失に動作するデバイスを目標としていることを考慮すると, この 3 者の中ではコア長 3mm が望ましい結果と言える.

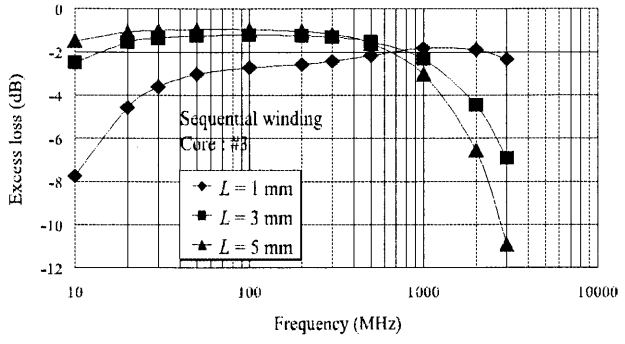

図 2.7: 損失にコア長の変化が与える影響 (計算値, 整合トランス,4 分配器)

次に透磁率の変化が伝達特性に与える影響を図 2.8 に示す.1GHz を超えるような高い周波数帯域では, 損失はどのモデルも大同小異である. 一方で,10MHz と言った低い周波数帯域では, その差は顕著ではないものの確かに存在し, 初期透磁率が高いものほど損失が小さく伝達特性が良好だと言える.

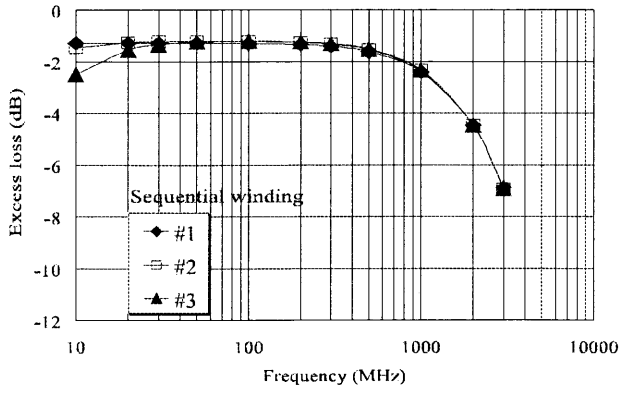

図 2.8: 損失にフェライトコア透磁率分散値の変化が与える影響 (計算値)

次は巻き線と伝達特性の関係である. 計算結果を述べる前に, 巻き線の構成について説明する. 図 2.1, 図 2.2, 表 2.1 で示したように,4 分配器に用いられる整合トランスの一次巻き線は 4 巻きである. 入力側から見て一次巻き線は 4 巻きのうち最初の 2 巻きは一次巻き線として動作する. そして残りの 2 巻きは一次巻き線と二次巻き線の二つの役割を果たす. 以後, 最初の 2 巻き (一次巻き線) をグループ 1 として, あとの 2 巻き (二次巻き線) をグループ 2 と表現する. 入力側から見た巻き線が 1234 の順でフェライトコアに巻きついている概略を図 2.9(a) に示す. このモデルは, 巻き線が入力側から順々にフェライトコアの周りに巻きついている配置から順巻き (sequential winding) と呼称する. 一方で, 入力側から見た巻き線が 1324 の順でフェライトコアに巻きついている概略を図 2.9(b) に示す. これは,2 番目の巻き線と 3 番目の巻き線と位置が順巻きと比較すると逆転した形である. これを交差巻き (alternative winding) と呼称する.

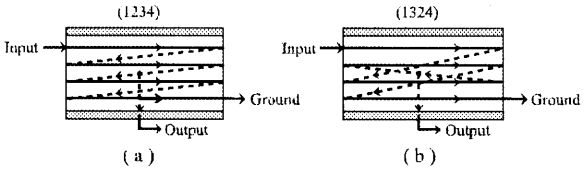

図 2.9: 巻き線の構成図 (a) 1234 (b) 1324.

損失特性を示している図 2.10 から分かるように, 交差巻き (1324) と交差巻き (1423) の構成では, 特に高い周波数で顕著な損失特性の改善が見られる.GHz 帯域での順巻きと交差巻きに関する過剰損失の注目すべき差は, その帯域ではフェライトコアの透磁率が 1 に近づき, 結果として順巻きでは一次側と二次側の巻き線を取り巻く磁束が効果的に鎖交せず, 結合度が悪くなる一方で, 交差巻きではグループ 1 とグループ 2 が物理的に鎖交しやすくなっている事によって引き起こされている事に起因している. 図 2.10 中の交差巻きで唯一順巻きより特性の悪い 2134 に関してであるが, その原因は, グループ 1 とグループ 2 が物理的に離れており, 電気的な結合が非常に弱くなったためと考えられる.

実際に高周波域では結合がどのようになっているかをシミュレーションを用いて調べた.2GHz での結果を結果を図 2.11 に,3GHz での結果を図 2.12 に示す. ともに図左が 2134 巻きの計算結果を表しており, 図右が 1423 巻きの計算結果である. 図からわかるように, 図右では全体的に磁界が分

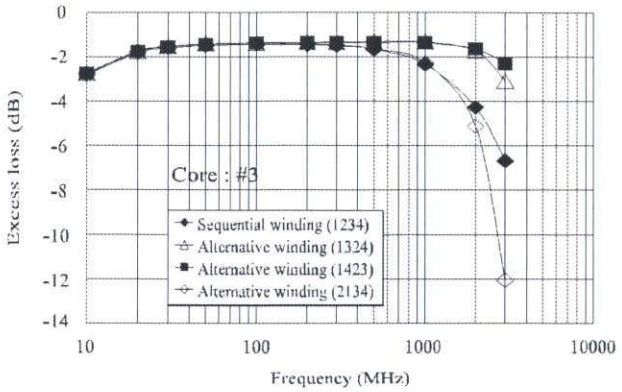

図 2.10: 損失に巻き線の巻き方の変化が与える影響 (計算値)

布している. しかしグループ間の結合でみると図左の方が良いことがわかる.

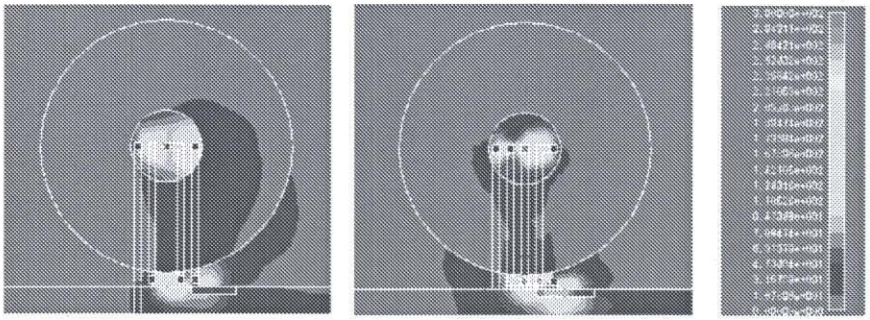

図 2.11: 交差巻き (2134) と順巻き (1234) の比較 (計算値,2GHz)

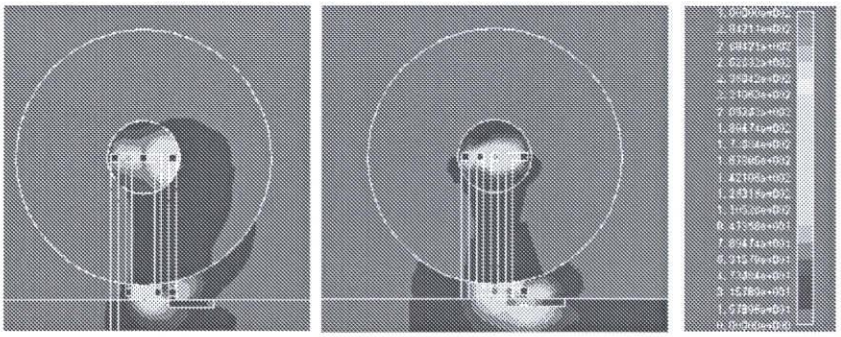

図 2.12: 交差巻き (2134) と順巻き (1234) の比較 (計算値,3GHz)

最後にフェライトコアの空洞部を上下にシフトさせて得られた結果を図 2.13 に示す. フェライトコア中心からの距離を b と定義した. 図中にある"upper"と"lower"の意味は, フェライトコア中心

より基板側によるものを"lower", 逆方向に偏芯したモデルを"upper"としている. 図 2.13 は偏芯量 b の変化と過剰損失をまとめた結果である. 偏芯構造によって高域での損失を低下させられる可能性のあることがわかる

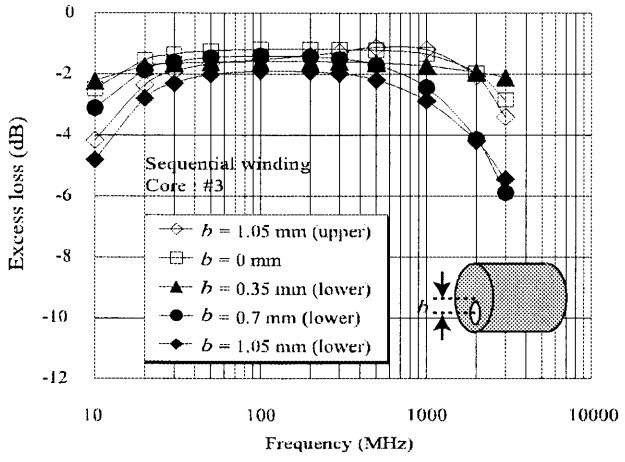

図 2.13: 損失にフェライトコア空隙部位置の変化が与える影響 (計算値)

2.4　実測

以上の計算結果, 特に巻き線の特性に与える影響の妥当性を検証するため, それぞれの種類のトランスを基板に実装しデバイスを試作した. 一例として, 順巻き (1234) を使用した整合トランス (T_3) の試作品とその等価回路をそれぞれ, 図 2.14(a) と図 2.14(b) に示す. この回路基板は整合トランス (T_3) を評価するためのものであり, 抵抗で構成されたインピーダンス整合回路 A と B がトランスの両端に配置されている. インピーダンス整合回路 A と B の損失特性を把握するために,A の回路を二つと B の回路を二つを鏡像対称結合した回路基板を試作した (図 2.15(a), 図 2.15(a)).A もしくは B 単体の回路損失を算出するには, 試作基板から得られた値を 2 で割ることで求めることが可能である.

これらの基板をネットワークアナライザーにつなぎ, 回路全体の損失と二つのインピーダンス整合回路 A と B をそれぞれ測定した. 結果を図 2.16 に示す. 順巻き整合トランス (T_3) 単体の損失を得るために, インピーダンス整合回路 A と B の損失を回路全体の損失から差し引くことを行なう. 測定結果を, 図 2.17 に示す. 図 2.17 には, 分配トランス (T_1) について同様の手法で測定した結果も合わせて載せている.10MHz から 3000MHz の帯域で, 分配トランス (T_1) の伝送損失は整合トラン

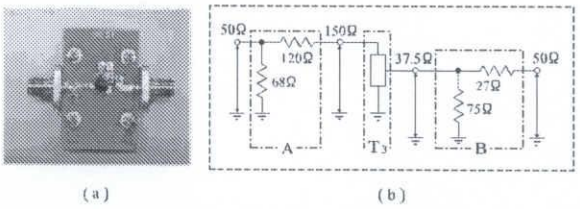

図 2.14: (a) 整合トランス評価用電気回路 T_3 (b) その等価回路

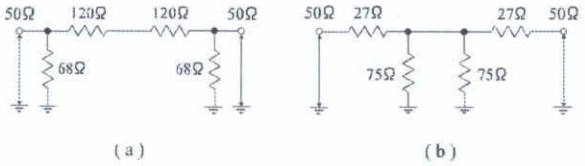

図 2.15: 整合トランス電気回路測定値 (a) A and (b) B

ス (T_3) に比べ無視できる程度である.

また, 反射特性 (リターンロス) についても評価を行なった.T_3 単体のリターンロスを直接測定し評価するのは, インピーダンスの不整合があるために困難である. 代わりに図 2.14(a) の回路を測定した. 図 2.14(a) の入出力面におけるリターンロスはそれぞれ,$24dB$ と $10dB$ より大きいことが測定できた. このことから T_3 単体のリターンロスは, これらの値よりはよいであろう事が言える.

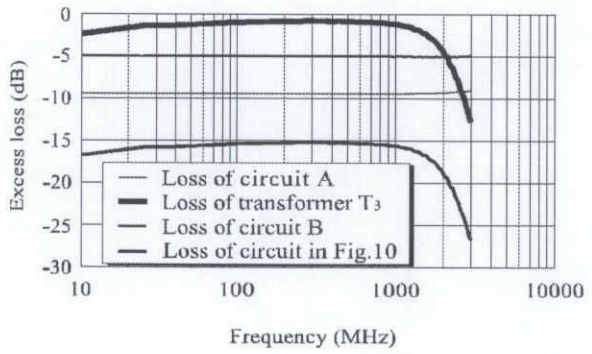

図 2.16: 図 2.14 における損失測定値とその整合トランス回路 A,B

次に 4 分配器の整合トランスにおける巻き線の構成が伝達特性に与える影響について検討した. 試験に使用する設定パラメータは, 長さ 3mm, 透磁率は表 2.2 の#3 の値とし, 偏芯はしていないフェライトコアを使用した. 結果は図 2.18 に示してあり, 比較のため同等のモデルを計算した結果

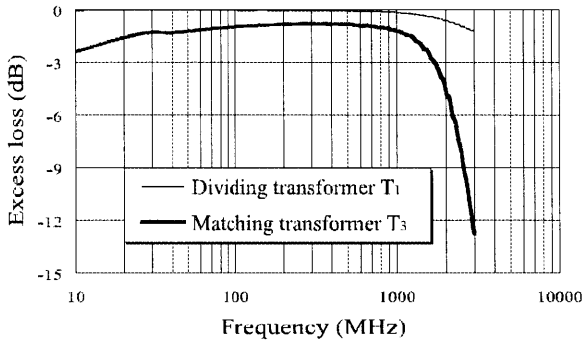

図 2.17: トランスにおける損失 (実測値, 分配トランス T_1 と 整合トランス T_3)

も併記した.

実験結果とそれに対応する計算結果との間には, いくらか不一致もある. これは計算誤差によるものの可能性が考えられる. しかしながら, 伝送特性に与える影響はどちらも似通っている. 言い換えれば, シミュレーション結果が提案したように, 実測でも, 特に GHz 帯域で, 交差巻きによって重要な損失特性を改善する要因となった. 日本の CATV システムに利用されている 10MHz から 2600MHz の周波数帯域において, 過剰損失は回路全体の損失から分配損失を差し引くことで $3dB$ 以下であることが分かる. 以前の CATV システム用 RF 分配器の過剰損失が同等の周波数帯域で $5dB$ を上回っていることを考えると, 本研究の結果は重要な改善と言える.

図 2.19 は偏芯量が b= 0.75mm(lower) であるフェライトコアの損失曲線を表したものである. 結果によれば, 以下のような事柄が言えるのではないかと思われる. つまり偏芯コアを用いることで GHz 帯域での改善が見られるという事である. それは図 2.13 で提案した通りである. ただし計算結果と実測には偏芯量に違いはあるが,lower に偏芯させることの優位性があることは双方とも一致した結果であった. また優位性が持てる周波数帯域は我々のターゲットとしている範囲 (2600MHz まで) より高い周波数である. しかし広帯域の動作周波数帯域を確保できると言うことは重要なことであると言える.

4 分配器で使用されるトランスの設定パラメータは表 2.1 の T_3 とで#3 とし, 巻き線の巻き方を交差巻き (1324) を採用したモデルを試作した. 図 2.20 は 4 分配器の試作機の写真であり, 図 2.21

はその伝達特性の測定値である.4 分配器全体の過剰損失は 10MHz から 2600MHz の周波数帯域で $3dB$ 以下であった. この結果は, 分配器の一部であるインピーダンス整合回路より少し小さい値である. その理由は, 分配器におけるトランスは巻き線によって結合しており, 一方で図 2.14 であるような整合トランスはマイクロストリップラインで結合している. その両者を比較した場合, インピーダンス不整合による結合損失が生じているためではないかと思われる. また入出力の反射損失は $11dB$ 以上, 出力間のアイソレーションは $25dB$ 以上であった.

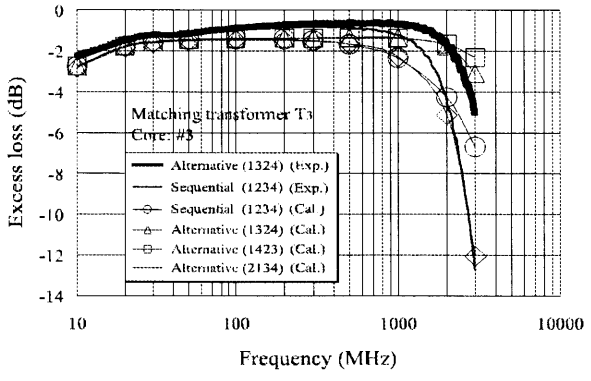

図 2.18: 整合トランスにおける損失 (実測値,4 分配器用, 交差巻き)

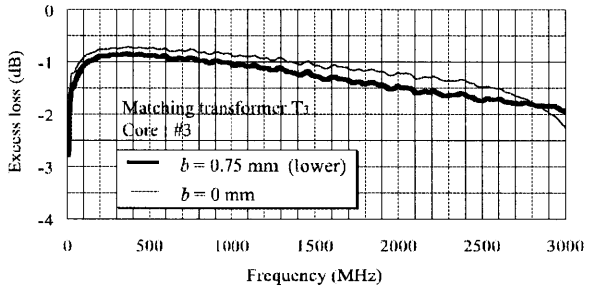

図 2.19: 損失にフェライトコア空隙部の変化が与える影響 (実測値)

2.5　まとめ

CATV 用 RF 分配器に用いられるトランスの伝達特性について解析を行った. 有限要素法を用いたコンピュータシミュレーションによって, 分配器の動作周波数を広帯域化させる目的で, 伝達特性とトランスの設定パラメータについて研究を行ってきた.CAE は複雑な形状のトランスをモデル化し, その伝達特性を評価することを可能にしてくれた. 今回の研究では, 巻き線の巻き方がトラ

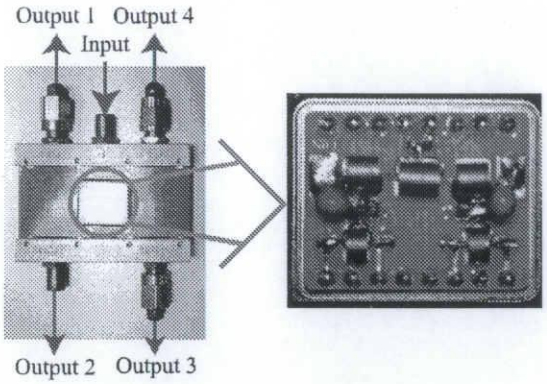

図 2.20: 試作 4 分配器

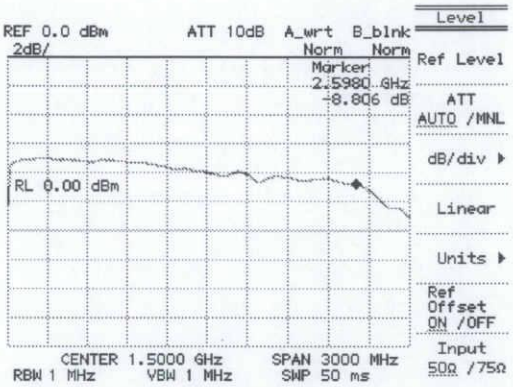

図 2.21: 整合トランスの伝達損失 (実測値,4 分配器, コア長 3mm, 透磁率分散#3, 交差巻き)

ンスの伝達特性に大きな寄与をしていることが分かった. 特に分配器で生じる損失の大きな部分を占める整合トランスでその傾向が顕著であった. また従来から言われていたことだが, 交差巻きのほうが順巻きに比べ高い周波数で良好な特性を得られると言う事実もシミュレーションを用いて確認することが出来, 超広帯域においてデータを得ることができた. シミュレーションに沿って試作したトランスを用い, これらの事実を検証した. 過剰損失は 10MHz から 2600MHz において $3dB$ を下回り, 動作周波数の広帯域化を示唆する結果を実測から得ることが出来た.

参考文献

[1] T. Kanie and T. Takeo: "Fabrication of CATV Broadband Splitter", The Transactions of The IEICE, Vol.J80-C-1, No.5 (1997), pp.250-251

[2] T. Kanie, T. Takeo, H. Ito, R. Hashimoto and T. Azakami: "Frequency-Range Widening of Three-Transformer CATV Divider Using a Bead Type Ferrite Core", The Transactions of The IEICE, Vol.J83-C, No.4(2000), pp.276-284

[3] T. Takeo, T. Kanie and H.Ito: "Fabrication of 4-Way splitter for CS Satellite N-SAT-110 Adaptive CATV systems", The Transactions of The IEICE, Vol.J85-C, No.3(2002), pp.206-207

[4] Y. Naito: "A Note on Permeability Dispersion of Spinel Ferrite", The Transactions of The IEICE, Vol.56-C, no.2(1973), pp113-120

2.6 付録

2.6.1 インピーダンス変換理論

分配トランス ($\mathbf{T_1}$ の場合)

入力電流は 2 分配され (出力電流の振幅は, 入力の半分), そして分かれた電流はそれぞれの巻き線中を反対方向に流れるので, フェライトコアの内部に磁束がほぼない状態になり, 二つの出力電圧は等しくなる. 結果的に, 出力側のインピーダンスは入力側に比べて二倍になる.

整合トランス ($\mathbf{T_3}$ の場合)

周知の通り, 入力 (Z_1) と出力 (Z_2) の理想的なトランスのインピーダンスは, 巻き線の一次側の巻き数 (N_1) と二次側の巻き数 (N_2) に比例し次式のようになる.

$$Z_1 : Z_2 = {N_1}^2 : {N_2}^2$$

整合トランス (T_3) では,$N_1 = n_5 + n_6$,$N_2 = n_6$ となる. それゆえ,$Z_1 : Z_2 = 16 : 4 = 4 : 1$ が得られ $Z_2 = Z_1/4$ であることが分かる.

第3章　光ファイバ波長分散二次歪み補償システムにおけるP-Pトランスの伝送特性解析

要約

光ファイバ伝送で発生する波長分散二次歪みを補償するための簡易な方式として提案されているプッシュプル伝送システムの性能を左右する重要な要因の一つであるプッシュプルトランスの伝送特性と，トランス構造との関係について電磁界シミュレーションによって詳しく検討した．最後にシステム全体の二次歪み改善性能についてもシミュレーションを用いて評価し実測と良好な一致を得た．

3.1　はじめに

P-P トランスの伝送特性の解析結果について述べる前に光ファイバ伝送において発生する波長分散二次歪みの補償システムについて説明する. 最近では光ファイバの普及は長距離通信伝送路すなわち幹線系のみにとどまらず，各家庭にも入り込もうとしており，いわゆるFTTH（Fiber to the Home）も現実のものとなりつつある．こうした伝送路の光化は，CATV（ケーブルテレビ）システムにおいても進められており，幹線系に光ファイバを利用したHFC（Hybrid Fiber Coaxial）と呼ばれるシステムが既に一般的になっている． CATV では，放送の通信との融合化やデジタル化などそれを取り巻く環境の変遷に伴って，設備投資の節約などを目的として市町村単位のネットワークを相互に接続する広域化も進められている．その際には，100km を越えるような長距離伝送も珍しくないことから，一般的な$1.31\mu m$波長光源では不十分であり，より低損失な$1.55\mu m$伝送も検討されている．ところが，周知のように，既設の$1.31\mu m$零分散ファイバにおいて$1.55\mu m$波長の光ファイバ伝送を行うと，波長分散のために歪み特性が大きく劣化する．分散を補償するためにとられる手法として, 下記のものが挙げられる.

1 分散補償ファイバを伝送距離に応じて適宜挿入する手法：

発生する波長の分散をキャンセル (補償) する特性を持つ光ファイバを使用する

2 光源の直接強度変調の代わって外部変調器を使用する手法 [1] :

入力光を外部信号によって, 位相, 振幅, 偏波面などを変化させ分散を補償する

3 光ファイバグレーティングなどを利用した可変型補償デバイスなどを用いる手法 [2]:

光ファイバグレーティングとは, ファイバコア上に周期的な屈折率変化を形成したもので, その周波数に応じた特定の波長のみを反射させる

しかしこれらの手法は, コストや損失, あるいはシステムに柔軟性がないなどの問題点も抱えている．これに対して，$1.55\mu m$ 波長を利用したときに生じるこの波長分散歪みを改善するための比較的安価で簡易な方式としてプッシュプル（P-P）伝送システムが提案されている [3]．本手法は，既存の設備を有効に活用できるという利点がある一方で，システムの非対称性による特性劣化が問題となる．そこで，本研究ではプッシュプル伝送方式の特性を左右する重要な要因の一つである P-P トランスの伝送特性について詳しく検討している．本論文では，P-P 伝送方式による二次歪み改善の基礎実験結果とともに，P-P トランスの構造と伝送特性との関係について電磁界シミュレーションにより検討した結果についても述べる．

3.2　光 CATV システムの伝送方式及び伝送品質

現在，ほとんどの CATV システムでは変調方式として半導体レーザ (LD) の直接強度変調方式を採用している．この方式では，多チャンネル映像信号やデータ信号などの周波数多重化された高周波信号 $i(t)$ を，バイアス電流 I_{op} に重畳した電流 $I(t)$ でレーザを駆動してレーザ光出力 P を強度変調し，光ファイバ長距離伝送のあとの受信側で，光信号を PIN-PD で光電変換して元の高周波信号を復元している．周知のように，LD を直接強度変調するとチャーピングにより LD スペクトルが広がる．特に，伝送波長として $1.55\mu m$ を利用する場合には，このチャーピングによる LD 光源のスペクトル拡散が波長分散に大きな影響を与える．直接強度変調における LD の駆動電流と光出力との関係を図 3.1 に示す．ここで，変調信号を

$$i = a * \sin(\omega t + \theta) \tag{3.1}$$

ただし a : 変調信号のピーク値, ω: 変調角周波数, θ: 初期位相 とする．このとき，光変調度あるいは変調指数 m は信号電流のピーク値 a によって,

$$m = \frac{a}{I_{op} - I_{th}} \tag{3.2}$$

で定義される．ただし，I_{th}, I_{op} はそれぞれ閾値電流及びバイアス電流である．

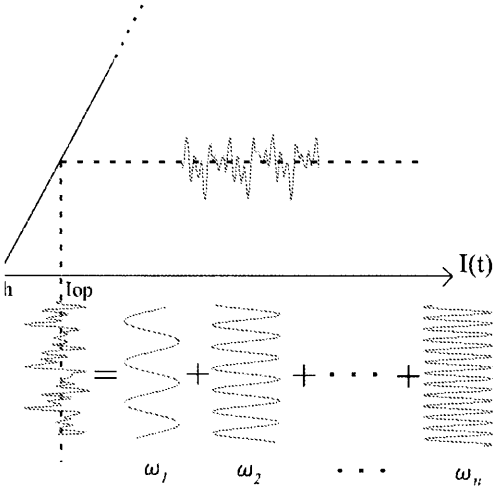

図 3.1: LD の直接強度変調

一般的に，CATV の伝送品質は，搬送波と雑音との比で定義される C/N と，多数のチャンネルの二次相互変調歪みに関係する複合二次歪み CSO（Composite Second Order Beat）や，三次相互変調歪みの集合体である CTB（Composite Triple Beat）などの歪み特性値によって評価される．雑音特性と歪み特性は相反関係にあり，C/N を大きくするために LD の光変調度を上げると CSO や CTB は劣化する．CSO は，伝送路内の多重反射などによっても劣化するが，特に 1.55μm 伝送では波長分散によって大きな影響を受ける．多チャンネル伝送における CSO は，関係する歪み周波数のすべての組み合わせに対する波長分散 2 次歪み (HMD2) を加え合わせたものであり，基本的に HMD2 と比例関係にある．本研究では前述の HMD2 に対して本方式による補償効果を評価対象としている．

3.3　波長分散二次歪み

波長分散による二次歪み発生の原因は, チャーピングの原理を理解することで説明できる. チャーピングとは, 半導体レーザにおいて, 高速で変調を行うと注入電流の変動に対応して屈折率の変化が生じ, 結果として波長 (周波数) が変化し, スペクトルが信号帯域以上に広がり, 分散を持つファイバの中を光パルスが伝搬する際に伝送特性の劣化が生じやすくなる現象を指す. チャーピングの概念を図 3.2 に示す.

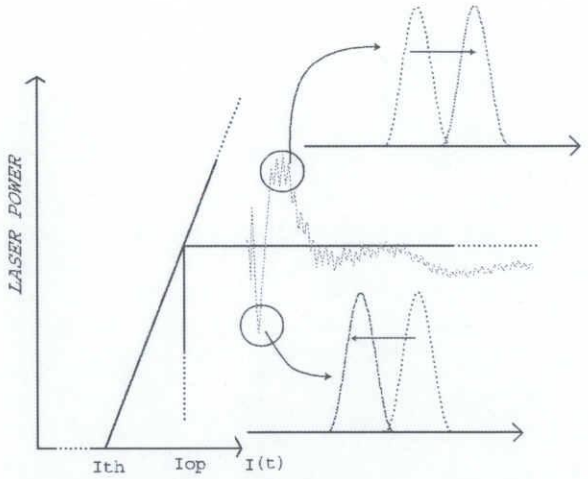

図 3.2: 光強度直接変調によるスペクトル拡散

この現象により変調電流振幅の大小に応じて発光波長が長短側へずれるため, 単一正弦波形を有する光信号は, ある距離にわたって光ファイバ伝送されると波長分散によって光強度の大小に応じて相対的に進んだり遅れたりして図 3.3 の左段上のように歪む. 具体的に言えば, 強度が強い光は, チャーピングにより波長は長くなり相対的に屈折率が小さくなる. この場合, 光は相対的に早くなる. また強度が弱い光は, チャーピングにより波長は短くなり相対的に屈折率が大きくなる. この場合, 光は相対的に遅くなる. この歪んだ伝送信号を周波数分解すると同図の左段下のようになり, 二次の高調波が発生することになる. 二次高調波歪み量は図 3.3 の右のように基本周波と 2 倍の周波とのレベル差として定義される.

チャーピングを考慮したときの波長分散による二次高調波歪みは次式のように表せる.

$$2HD_{DSP} = \frac{1}{4}m\ddot{\beta}z\Omega\sqrt{(4\gamma)^2 + (\ddot{\beta}z\Omega^3)^2} \tag{3.3}$$

ただし各変数は以下のように定義する.

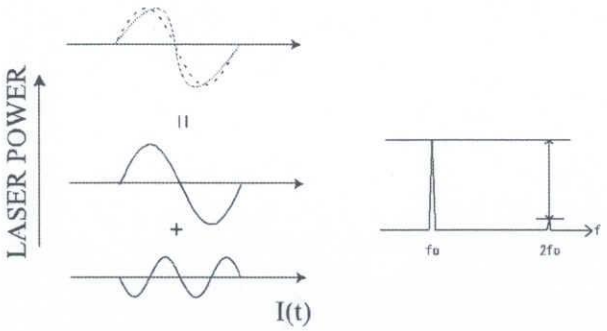

図 3.3: 波長分散二次歪みの計算値

m ： 光変調指数

$\ddot{\beta}$ ： $-\dfrac{\lambda^2}{2\pi c}D$($\lambda$: 光源の波長, c : 光速, D : 波長分散)

z ： 伝送距離

Ω ： 変調周波数

$\gamma = G(I_{op} - I_{th})$ ： チャーピング量 (G : レーザの周波数変調効率)

この式に基づいて，代表的なパラメータ値に対して二次高調波歪みを計算すると図 3.4 のようになる．ここで，光変調指数 $m = 0.1$〜1.0,$\lambda = 1.55\mu m$,$D = 17ps/nm/km$,$\Omega = 91.25MHz$,$G = 150MHz/mA$,$I_{op} = 24mA$，$I_{th} = 8mA$ を仮定した．

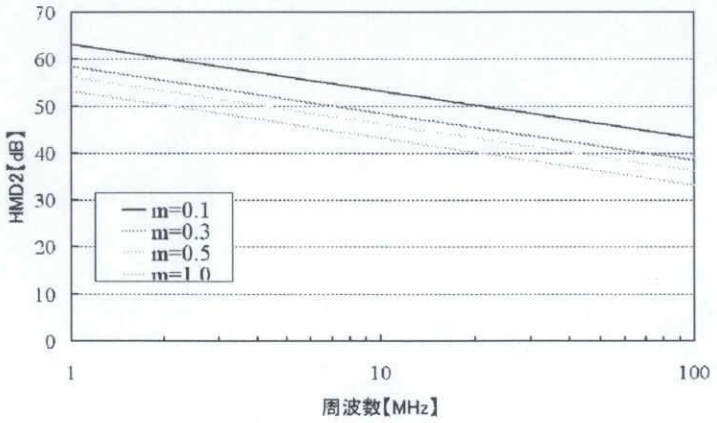

図 3.4: 二次高調波歪み量の計算結果

3.4　P-P 伝送方式による二次歪み補償

3.4.1　二次歪み補償システムの概要

文献 [3] では，こうした 1.55μm 波長伝送における二次歪みを抑制するための手段として，図 3.6 に示すような 2 つのレーザと光伝送路を利用した P-P 伝送方式が提案されている．この方式の基幹をなすアイディアの部分を図 3.5 に示す. この P-P 伝送方式では，高周波信号入力が増幅回路及び利得調整回路によって適切なレベルに調節され，そのあと P-P トランスを用いて，相互に位相が反転した二つの高周波信号に分けられる．これら二つの信号で半導体レーザ (LD1，LD2) を直接強度変調して，光ファイバ伝送を行う．光ファイバ出力端では PIN-PD によって信号を光電変換し，P-P トランスによって相互の位相を反転して二つの信号を再び合成する．波長分散による歪み発生原理に関して述べたように，光ファイバ伝送後は，LD1 からの信号の歪み波形と LD2 からのそれはちょうど逆の形に歪むため，受信側で位相反転して合成すると二次歪みがキャンセルされる. システム全体は図 3.6 のような構成になる.

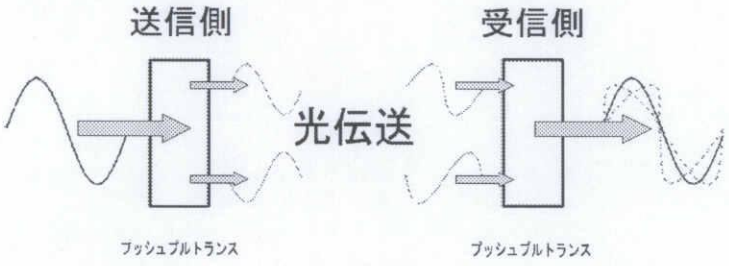

図 3.5: プッシュプル (P-P) 伝送方式の概念図

3.4.2　実験結果

図 3.6 のブロック図に基づいて試作したプッシュプル光伝送装置の特性を測定した．半導体レーザ（LD）としてはDFB-LD(標準出力 $2mW$，駆動電流 $24mA$，閾値電流 $8mA$，発振波長 $\lambda = 1.5514$ 及び $1.5504\mu m$) を使用した．また，光ファイバ伝送路は $1.31\mu m$ 零分散ファイバを用いた．$\lambda = 1.55\mu m$ におけるその波長分散値は $17ps/km/nm$ である．LD 駆動部分には抵抗素子によるインピーダンス整合回路を利用している．光伝送路における反射を低減するために，光コネクタとしては斜め球面研磨（APC）を用いた．以上の装置を用いて，P-P 伝送時の二次高調波歪みを図 3.7 の測定系によって測定した．図 3.8 は $20km$ の光ファイバ伝送において，P-P 伝送時と，通常の単一 LD 伝

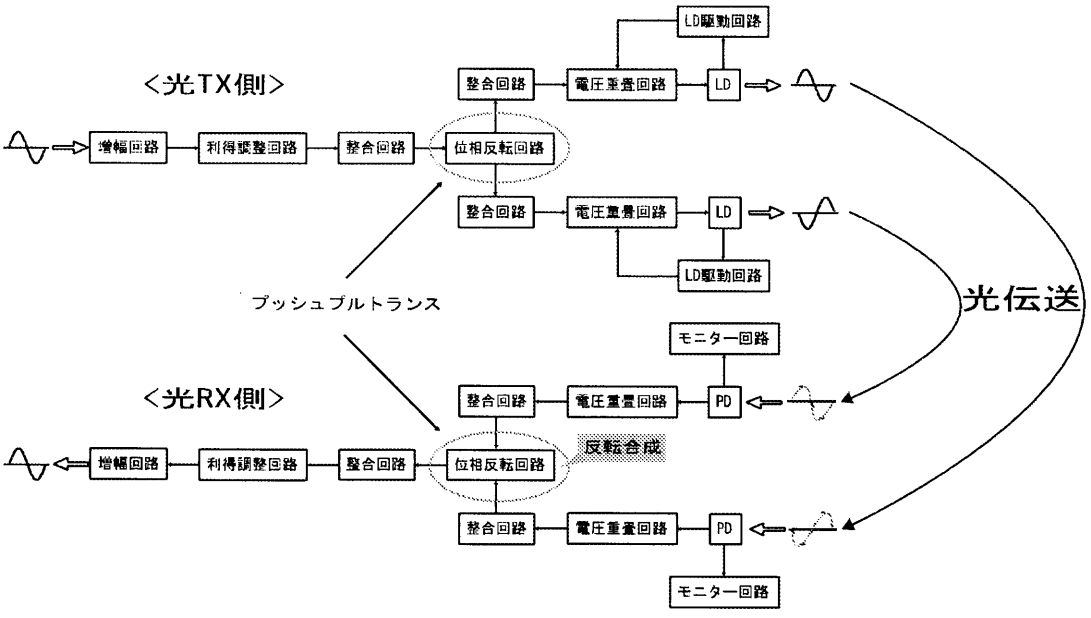

図 3.6: プッシュプル (P-P) 伝送方式のブロック図

送を行ったときの二次歪み測定結果の一例を示しているが，P-P 伝送によって歪み特性が約 $15dB$ 改善していることがわかる．なお，受信用 PIN-PD を含めた光リンク全体の周波数特性を測定した結果，変動は周波数範囲 10〜$50MHz$ においては $\pm 1dB$ 以内，また 10〜$500MHz$ においては $\pm 1.5dB$ 以内であった．

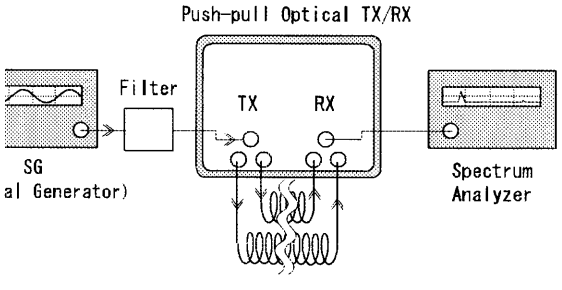

図 3.7: 二次歪みの測定系

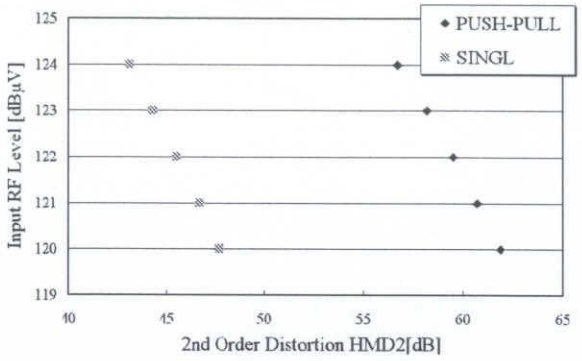

図 3.8: 二次歪み改善効果の測定結果

3.5　電磁界シミュレーションによるP-Pトランスの特性評価

3.5.1　解析モデル

以上，P-P光伝送方式によって二次歪みを補償できることを示す基礎データについて述べた．以下ではシステム全体の特性に大きな影響を与えるP-Pトランスの構造とその伝送特性との関係を電磁界シミュレーションによって詳しく調べる．図3.9にはP-Pトランスの回路を示す．P-Pトランスは送信側及び受信側に対の形で配置されているが，送信側と受信側では構造的に逆の形で用いられる．図3.9では，ポート1からの入力はポート2，3から互いに位相反転した信号として出力され，ある距離にわたって光ファイバ伝送されたあと，ポート3の信号は受信側P-Pトランスのポート2に，また送信側のポート2の信号は受信側P-Pトランスのポート3に入力される．送信側P-Pトランスの特性としては，理想的にはポート2，3から得られる出力信号の振幅は等しく，位相は反転していることが望まれる．受信側は送信側の逆の動作となるはずである．

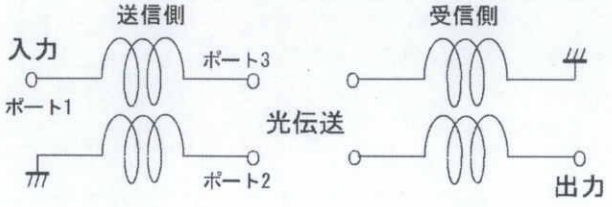

図 3.9: P-Pトランスの回路構成

上述したP-Pトランスの動作，ひいてはP-Pトランスを用いることで得られる伝送特性を有限要素法に基づく電磁界シミュレーションによって評価した．図3.10は計算に使用したP-Pトランスのメッシュモデルである．入出力はマイクロストリップ線路で構成されており，トランス部分

は外径 D_{out}，内径 D_{in}，長さ L の円筒形状のフェライトコアに 1〜2t の巻線が巻かれている．高周波トランスの研究で得た知見を元に, 計算ではフェライトコアの長さ，透磁率，巻線の巻き方や巻き数と伝送特性との関係について検討した．

検討パラメータ

1 フェライトコアの巻き線の巻き方

2 フェライトコアの長さ

3 フェライトコアの透磁率の変化

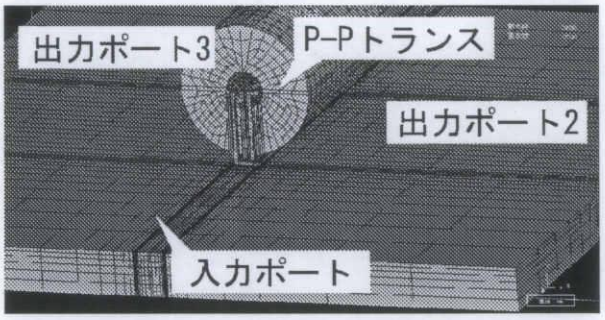

図 3.10: P-P トランス解析モデル

3.5.2　P-P トランス伝送特性の解析結果

まずは,P-P トランスにおいての巻き線の伝達特性に対する影響を検討した. 検討対象は,$L = 6mm$, 外径 D_{out}，内径 D_{in} とし, 計算周波数は 10MHz,500MHz,3000GHz とした. 磁界強度の計算結果をコンター図の形で,10MHz から順に図 3.11, 図 3.12, 図 3.13 に示す. 図中の番号は巻き線の番号である.1,2 番が 1 次側であり,3,4 が 2 次側である. 図から順巻きのほうが全体的な磁界強度は強いと思われるが, 交差巻きのほうが広帯域において 1 次側と 2 次側間の結合が強いことが分かる.

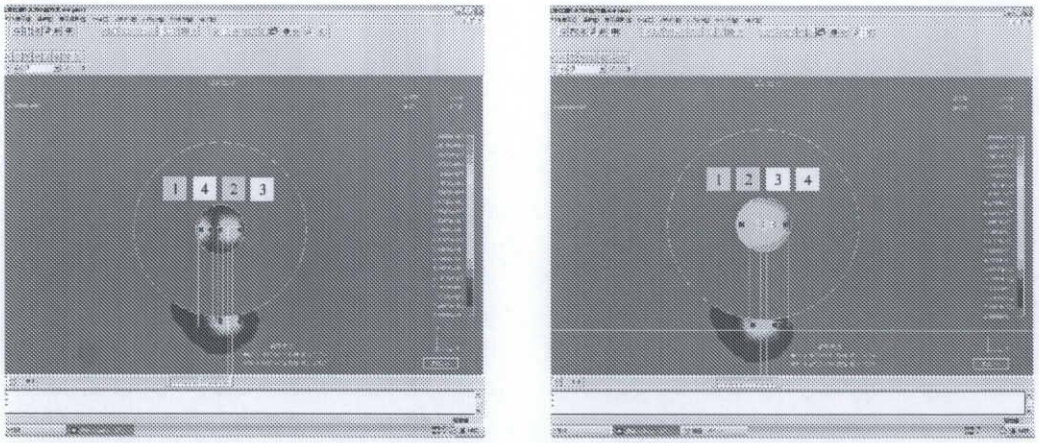

図 3.11: 6mm10MHz の磁界コンター図 (左) 交差巻き (右) 順巻き

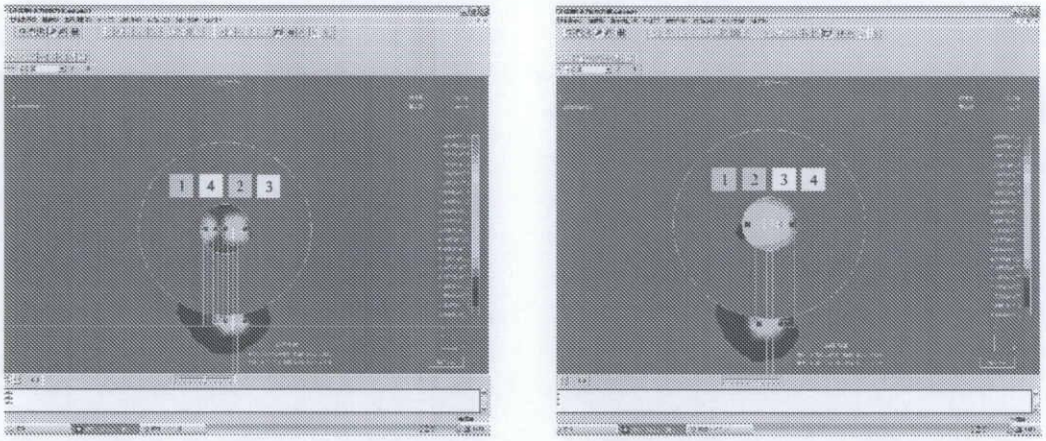

図 3.12: 6mm500MHz の磁界コンター図 (左) 交差巻き (右) 順巻き

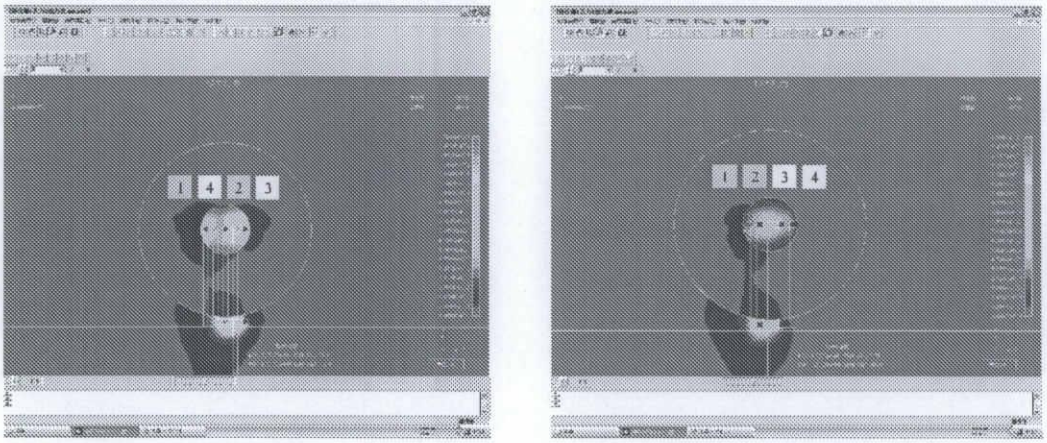

図 3.13: 6mm3000MHz の磁界コンター図 (左) 交差巻き (右) 順巻き

次に図 3.14 及び図 3.15 は $L = 6mm$, 巻線 $2t$ の場合に対する伝送特性の計算結果を示してお

り，図3.14はS_{21}あるいはS_{31}で表した入出力間の振幅伝達特性，図3.15は出力間の位相差である．このとき，トランス巻線の巻き方として，1次側と2次側を入力側から順々に巻く順巻きと，両者を交差させて巻く交差巻きの二通りの方法について検討した．入力信号が二つの出力に分かれるため，図の振幅伝達特性は$3dB$の分配損失を含んだ形になっている．入力に直結したポートであるポート3の方が，入力と反対側のポートであるポート2よりも，伝達特性は良好であることがわかる．また，周波数特性についてみると，$10MHz$〜$400MHz$の範囲ではほぼ平坦であるが，2tの場合，1GHzを越える周波数帯域では伝達特性は急激に低下してきている．一方，二つの出力間の位相差を示す図3.15をみると，$10MHz$〜$1GHz$ではほぼ180度の反転特性が維持されているが，やはり$1GHz$以上になるとその関係が崩れてくることがわかる．巻線の巻き数及び巻き方についてみると，巻き数に関しては，$1GHz$以下では$2t$が$1t$よりも優れ，$1GHz$以上では逆転する．$2t$の場合の巻き方に関しては，$1GHz$以上において交差巻きの方が伝達特性が良好である．

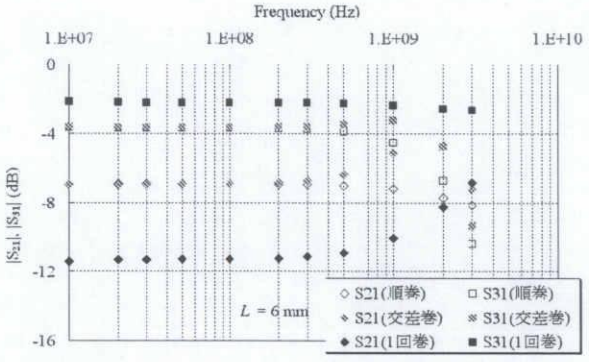

図 3.14: P-P トランスの振幅特性 (巻線依存性)

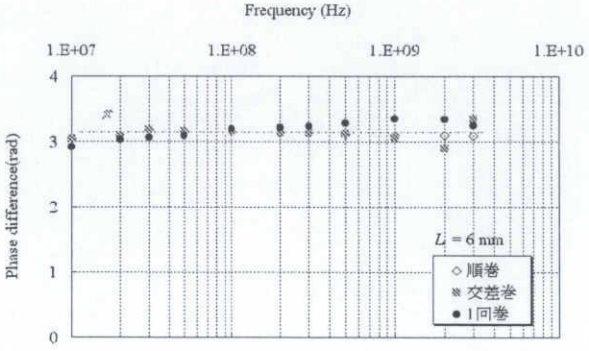

図 3.15: P-P トランスの位相特性 (巻線依存性)

次に，フェライトコアの長さと伝送特性との関係について調べた．その結果が図3.16及び図3.17である．$1GHz$以下ではコアが長いほど二つの出力の振幅特性は均一になるが，$1GHz$以上の周波数になると逆に短いコアの方が特性が良好である．位相特性に関しては，$1GHz$以下では大きな違いはないものの，$1GHz$を越えると短いコアの方が好ましいことがわかる．

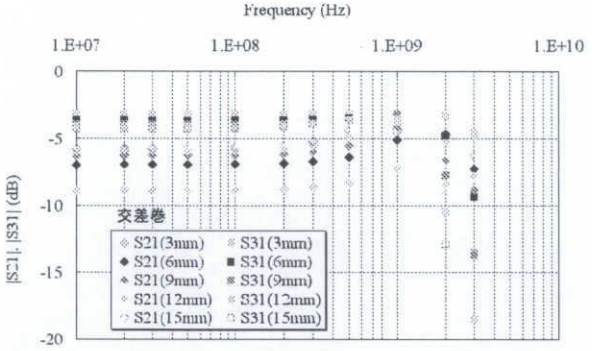

図 3.16: P-P トランスの振幅特性 (コア長さ依存性)

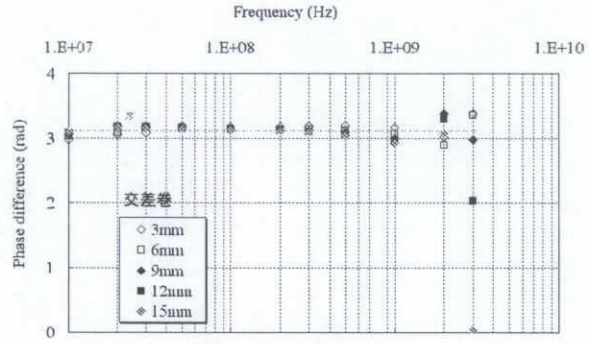

図 3.17: P-P トランスの位相特性 (コア長さ依存性)

さらに，フェライトコアの透磁率と伝送特性との関係についても調べた．この場合にも$1GHz$付近を境に傾向が分かれるが，$1GHz$以下では透磁率が大きいほど振幅，位相とも特性が良好であり，$1GHz$を越えると逆の関係がみられる．

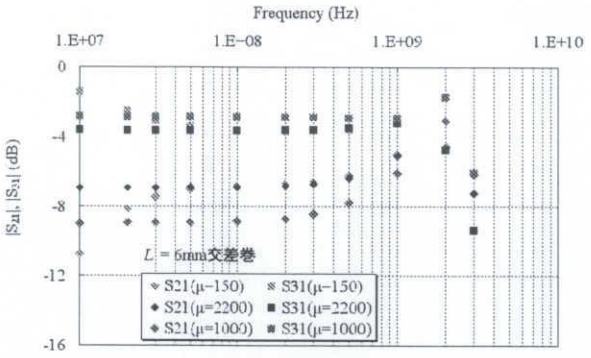

図 3.18: P-P トランスの振幅特性 (透磁率依存性)

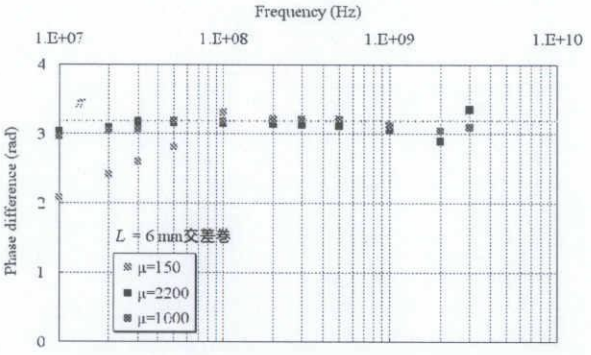

図 3.19: P-P トランスの位相特性 (透磁率依存性)

3.5.3　P-P 伝送システム全体の歪み改善性能評価

以上，P-P トランス単体の伝送特性を電磁界シミュレーションによって評価した結果について述べた．最後に，これらの結果をもとに，P-P 伝送システム全体の二次歪み改善性能について，同様にシミュレーションによって評価した．その結果の一例が図 3.20 である．図における周波数は信号基本周波数を表しており，例えば $200MHz$ の周波数における二次歪み量は，その二次高調波の周波数である $400MHz$ とのレベル差 (二次歪み量) の値を示している．計算では，P-P トランスとして $L = 6mm$ の $2t$，交差巻きのデータを，また光ファイバ伝送距離としては $20km$ を想定した．比較対象は,P-P トランスを使用せず $20km$ 光ファイバ伝送した時の二次歪み量とした. 計算方法と評価量に関して次に述べる.

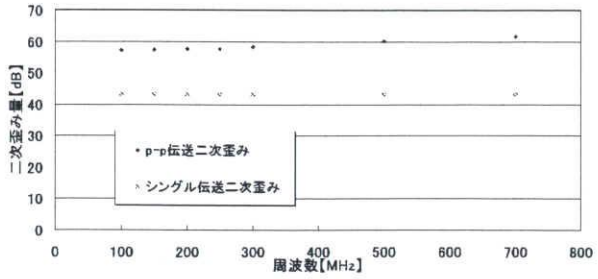

図 3.20: 二次歪み改善効果の計算結果

計算方法

前述したP-Pトランスの説明と重複するが,P-Pトランスの動作を解説しながら計算方法について図3.21を用いながら述べる.

信号が入ってくる(1.から入力)と,点線Xの部分で振幅A倍で位相がα変化した信号(3.から出力)と振幅B倍で位相がβ変化した信号(2.から出力)に分かれる.理想的には,$A = B = 0.5$,$\alpha = 0[rad]$,$\beta = \pi[rad]$であるが,そうならないことはすでに示したとおりである.信号が二つに分かれた後,半導体レーザ(LD1, LD2)を直接強度変調して,光ファイバ伝送を行う.光ファイバ出力端ではPIN-PDによって信号を光電変換し,P-Pトランスに2つの信号を入力する(2.と3.から入力).そして最後に出力を得る(1.から出力).受信側(RX)での動作は,送信側(TX)のそれと同様である.

二次歪み量は基本波と二次高調波のレベル差として定義しているが,比較対称となる通常の手法(単一伝送路)で信号を送った場合の二次歪み量は,光ファイバ内での距離減衰と波長分散二次歪みを考えればよい.上記の条件で測定した結果,約$40dB$であった.一方,P-Pトランスを用いた場合は次のような手順で二次歪み量を求める.まず基本波と二次高調波の振幅を計算し,その数値を元にソフトウェア(octave)を用いてその振幅を有する信号を作成する.次に2つの信号を足し合わせ,FFTを行いレベル差をデシベル変換することによりデータを得ることが出来る.そのソフトウェアはoctaveを用いて自作した.コードは付録を参照されたい.また基本波と二次高調波の振幅を計算方法であるが,図3.21において点線Y部分での振幅をCで表すと,次式のとおりである.

基本波の振幅

$$\sqrt{\{(A*B)*(cos(\alpha+\beta))\}^2+\{(A*B)*(sin(\alpha+\beta))\}^2} \tag{3.4}$$

二次高調波の振幅

$$\sqrt{\{(A*B*C_{23})*(cos(\alpha+\beta))\}^2+\{(A*B*C_{32})*(sin(\alpha+\beta))\}^2} \tag{3.5}$$

ただし図中において経路2.→3.の時のCをC_{23}とし,経路3.→2.の時のCをC_{32}と表記している.このように表記する理由は,P-Pトランスの非対称性によりそれぞれの経路内のレーザー強度が異なり,結果としてチャーピング量が各々違うため,C_{23}とC_{32}の値が異なるためである.しかしながら今回は簡単の為,それぞれを等しいと仮定した.

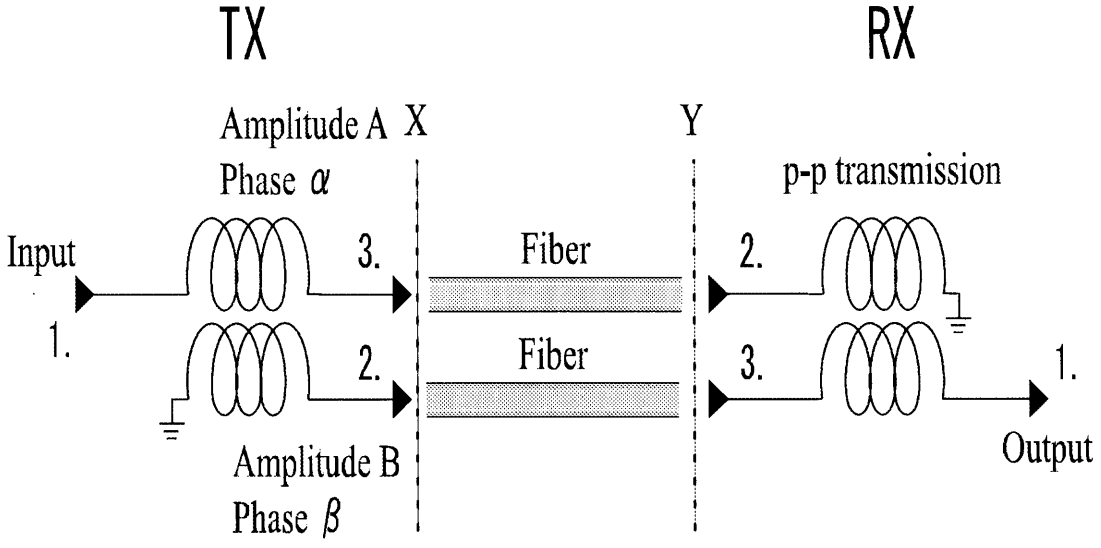

図 3.21: P-Pトランスの二次歪み補正方法概略

結果と課題

図3.20からわかるように$700MHz$までの周波数範囲にわたって約$15dB$の改善効果が得られるという結果を得ることが出来た.これは実測した結果によく一致している.このことから有限要素法を用いたシミュレーションの妥当性を言うことが出来ると思われる.

シミュレーション結果からは,P-Pトランスの理想としている$A=B=0.5,\alpha=0[rad],\beta=\pi$は得ることが難しいように思われる.そこで,今後の課題であるが,P-Pトランスは振幅と位相が異なった非対称性は完全になくせないと仮定し,非対称性と伝送特性との関係を明らかにする必要がある.

また本手法は, 光ファイバを2本使用して伝送を行なう. これはつまり光ファイバの長さに違いが生じ, チャーピング量がそれぞれのファイバ内で異なると言った結果になりえる. そこで,2本のファイバ長の違いなどを自作のoctaveソフトに組み込んだ計算なども考えている. その際に使用するチャーピング量を式3.3から設定できるようにする予定である.

3.6　まとめ

光ファイバ伝送において$1.55\mu m$波長を用いたときに発生する波長分散二次歪みを補償するための安価で簡便な方式として，プッシュプル伝送方式を検討し，その基礎的な改善特性について実験的に確認するとともに，その補償性能を支配する要因の一つとしてP-Pトランスの伝送特性をシミュレーションによって評価した結果について述べた．今後は実験及びシミュレーションの両面においてさらに詳しい検討を加え，本システムの二次歪み改善性能について明らかにする予定である.

参考文献

[1] M. Moshe, J. Berger, A.J. Ley, I.M. Levi and Y. Kagan, "Progress in Externally Modulated AM CATV Transmission Systems," J. Lightwave Technol.,Vol.11,No.1(1993)

[2] 磯野秀樹, "分散補償デバイスの現状と展望," レーザー研究, Vol.30, No.10(2002)

[3] 竹尾隆, 蟹江知彦, "光CATVシステムにおける波長分散二次歪み補償装置の試作," 電子情報通信学会論文誌, Vol.J88-C, No.7, PP.574-575, 2005

[4] 加藤弘晃, 野呂雄一, 竹尾隆, 蟹江知彦, 小田究, 伊藤治彦, "光CATVにおけるプッシュプル式波長分散二次歪み補償装置の特性評価," 第36回光波センシング技術研究会講演論文集, PP.135-140, 2005

[5] M.R.Phillips, T.E.Darcie, D.Marcuse, G.E. Bodeep, and N.J.Frigo, "Nonlinear Distortion Generated by Dispersive Transmission of Chirped Intensity-Modulated Signals," IEEE Photon. Technol. Lett., Vol.3,No.5(1991)

3.7　付録

本研究で自作したoctaveソースコードを記載する. このソフトウェアの特徴は下記の通りである. 振幅の異なる2つの信号とその二次高調波を足し合わせ一つの信号とし,FFTをかけてその信号の周波数特性を得る. またその際位相を0から2πまで変化させることが可能であるため, どのような非対称性ならばP-Pトランスの機能として良好な特性を出せるかを検討することが可能である. また距離減衰に関しても, このプログラム内で組み込むことが容易に可能である.

■■■ original_renew.m ■■■

```
clear;
% ■ 計算時間測定開始 ■ %
% tic
% ■ 波形データとFFTデータの算出 ■ %
% fileout=fopen("***.txt","a");
%　出力ファイル名% filetmp = load 入力設定ファイル名.dat % これを使用し fs,f1,f2,amplitude を決める
fs = 3000; % サンプリング周波数
f1 = 200; % 信号周波数1
f2 = 400; % 信号周波数2
n1 = 300; % 信号長 (fs/f1(と fs/f2 だがここでは周波数が2倍なので必要ない) の整数倍にする)
n2 = 1000; % θの分解能 (θを何等分するか)
f = fs/n1*[0:n1/2-1]; % 周波数軸 (0～fs/2まで)
x = 1;
y = 1;
main1_amplitude = 0.5; % 第一信号 (基本波) の振幅
sub1_amplitude = 0.005; % 第一信号 (高調波) の振幅
main2_amplitude = 0.5; % 第二信号 (基本波) の振幅
sub2_amplitude = 0.005; % 第二信号 (高調波) の振幅
```

```
for i = 0:1:n1-1
  for theta = 0:1:n2-1
    % 第一信号 %
    a = main1_amplitude*sin(2*pi*f1/fs*i)+sub1_amplitude*sin(2*pi*f2/fs*(-i));
    % 第二信号 %
    b = main2_amplitude*sin(2*pi*f1/fs*i)+sub2_amplitude*sin(2*pi*f2/fs*i+2*pi/(n2)*theta);
    c = a + b;
    k(y,x) = c; % k(行, 列) に c の値を代入
    y++;
  endfor
  x++;
  y = 1;
endfor
for o=1:1:n2
  d=k(o,:); % d(?,?) は k(?,?) の 1 行目だけを代入
  e=fft(d);
  % 行 (横):一行でひとつの時間波形の FFT をかけたデータ
  % 列 (縦):一列である時刻の位相差の変化のデータ
  wav(o,:)=abs(e(1:n1/2));
endfor
% ■ HMD2 算出のためにスペクトラムのピーク値を求める ■ %
for y = 1:1:n2
  for x = 1:1:(f1*n1)/fs+1
    % 1～where+1+1 まででのピーク値
    peak_search1(x)=wav(y,x);
  endfor
  for x=(f1*n1)/fs+2:1:n1/2
    % where+1+2～f2 まででのピーク値
```

```
    peak_search2(x)=wav(y,x);
  endfor
  peak1 = max(peak_search1);
  peak2 = max(peak_search2);
  peak(y,1) = peak1;
  peak(y,2) = peak2;
  peak_search1 = 0;
  peak_search2 = 0;
endfor
% ■ dB を求める ■ %
tmp=zeros(1,n2);
for xxx=1:1:n2
  peak_tmp1 = peak(xxx,1);
  peak_tmp2 = peak(xxx,2);
  tmp(1,xxx) = (peak_tmp1)/(peak_tmp2);
endfor
% 横軸をラジアンに変更する
rad = linspace(0,2*pi,n2);
% 最大値を dB 単位で求める
dB = abs(20*log10(tmp));
% ■ 終了作業 ■ %
data = [rad;dB]';
save -append -ascii make_sure.txt data
% status=fclose(fileout); % ファイルクローズ (ファイルをオープンしていた場合)
% 計算時間測定終了 %
% toc
```

第4章　C言語による汎用FDTDプログラムの開発

要約

近年の電磁界数値解析の分野において, 有限要素法, モーメント法等の各種数値計算法が極めて大きな威力を発揮し, 複雑な形状や構造を持った電磁環境の解析が行なえるようになっている. このような背景をもとに, 従来の差分法を時間領域まで拡張したFDTD(Finite Difference Time Domain)法, つまり時間領域差分法が各種問題の解析に応用されている. この方法の最大特徴は, 最終的な結果やその結果に至るまでの電磁界の状況を視覚的に観察できることである.

本研究では, 第2章及び第3章で述べたような高周波デバイスの時間応答特性を解析する為にFDTDに基づくプログラム開発を並行して行なった. プログラム言語にはC言語を用い, 文献[1]に従いFDTDのプログラミングをした. プログラミングにあたっては, 公開されているいくつかのソースを参考にしたが, それらは基本的にFORTRANにより記述されておりC言語に移植しなければならない. そこでC言語による具体的なプログラミング方法に関して検討し, まとめた結果を報告する. まずはFDTDとは何かについて文献[1]を元に解説し, 続いてFDTDの中でもプログラムが難しいとされる境界面に関して説明を行なう. 最後に付録として, 作成したソースコードを記載する.P-Pトランスなどへの応用は今後の課題である.

4.1　はじめに

FDTDに関しての詳細は, その文献を参考にしていただくとして, 簡単に述べさせていただければ, 次の通りである.

FDTD法 (Finite Difference Time Domain method) とは, マクスウェルの微分方程式を差分し, 時間領域で解く方法である. この手法は, モーメント法や有限要素法などとともに数値電磁解

析 (computational electromagnetics) の分野における主要な解析手法と言ってよく, 近年多数の論文が提出されている. 時間領域を差分することにより時間変化も追える, と言うことが非常に強力なツールとなりえた理由の一つである.

しかし問題点として, 吸収層が独特の扱いとなる点とプログラムソースに関して, ほとんどの記述はFORTRANで記述されてある点である. 私の研究室でもそうであるがC言語での資産は世の中に多くあり, それを生かしプログラムを組むには,FDTDのソルバーもCで記述されてあることが好ましい. もちろんFORTRANも状況としては似ているが, 現在C言語の開発適応範囲の増加を考えれば,C言語で組むメリットは大きい. しかしながらC言語など他の言語でFDTDをプログラムしたソースはネットおよび文献等にもほとんど公開されていない. また公開されていたとしても, 空間サイズが可変であるなど汎用性のあるものは見当たらない. そこで本研究では, 以上のことを念頭に置きC言語でプログラムを組んだ. 以下では,FDTD法のアルゴリズムに沿って作成プログラムの概要や留意点をまとめ, 最後にソースを添付する.

4.2　FDTDの基本概念と定式化

4.2.1　マクスウェルの方程式

FDTDの根幹を成すマクスウェルの方程式を簡単にまとめる. 電界を$E[V/m]$, 磁界を$H[A/m]$, 電束密度を$D[C/m^2]$, 磁束密度を$B[T]$とし, その源である電荷密度, 電流密度をそれぞれ$\rho[C/m^2]$,$J[A/m^2]$とすると, 微分形式のマクスウェルの方程式は次式のようになる.

$$\nabla \times \vec{E}(\vec{r},t) = -\frac{\partial \vec{B}(\vec{r},t)}{\partial t} \tag{4.1}$$

$$\nabla \times \vec{H}(\vec{r},t) = \frac{\partial \vec{D}(\vec{r},t)}{\partial t} + \vec{J}(\vec{r},t) \tag{4.2}$$

$$\nabla \cdot \vec{D}(\vec{r},t) = \rho(\vec{r},t) \tag{4.3}$$

$$\nabla \cdot \vec{B}(\vec{r},t) = 0 \tag{4.4}$$

FDTD法において使用するのは, ファラデーの法則とアンペアの法則の2式である. よってガウスの法則は基本的に使用しない. マクスウェル方程式の積分形式で表記することは可能であるし,

それを用いてFDTDを定式化することももちろん出来る. しかしFDTDのそれは, 一般的にマクスウェル方程式の微分形を用いるので, 本論文では微分形式で話を進める.

4.2.2　Yeeアルゴリズム

FDTD法では, 波源または散乱体を囲むように解析空間をとり, 解析空間全体を微小な直方体 (以下セルと省略する) で分割する. 次に, 全セルに対して微分形式のマクスウェルの方程式を適用し定式化する. その基本なアルゴリズムとしてYeeアルゴリズムがある. 本節ではYeeアルゴリズムについて簡単にまとめる.

ファラデーの法則とアンペアの法則を時間差分する際,FDTD法では1次の差分公式が用いられる.1次差分には前進差分, 中心差分, そして後進差分があるが, 中心差分の精度は,$(\Delta)^2$のオーダーである. そして中心差分が最も精度がよいことからFDTD法では, 中心差分を用いるのが一般的である.

中心差分を採用することによって, 電界と磁界は時間的に交互に配置されることになる. 図4.1のように, 電界Eを$n=$整数次の時刻に, 磁界Hを$n=$半奇数次の時刻に割り当てることとする. 実際の計算においては, ある時刻の電界E^nを計算する際には1ステップ前の時刻の電界E^{n-1}と半ステップ前の時刻の磁界$H^{n-1/2}$から算出する. ある時刻の磁界$E^{n+1/2}$も同様に,1ステップ前の時刻の磁界$H^{n-1/2}$と半ステップ前の時刻の電界E^nから算出する.

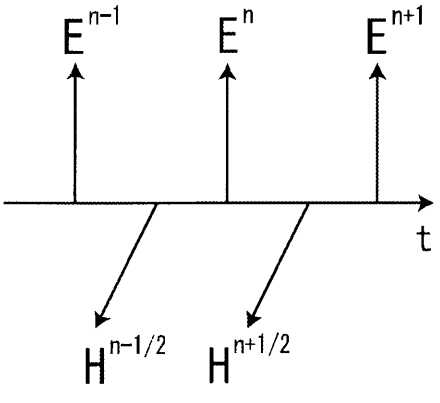

図 4.1: 電磁界の時間配置

現行のモデルではプログラムの簡単のために媒質は等方, 非分散性としている. その条件の元, ファラデーの法則とアンペアの法則を構成方程式 $B = \mu H, D = \epsilon E, J = \sigma E$ を用いて電界, 磁界に直すと次式のようになる.

$$\frac{\partial \vec{E}}{\partial t} = -\frac{\sigma}{\epsilon}\vec{E} + \frac{1}{\epsilon}\nabla \times \vec{H} \tag{4.5}$$

$$\frac{\partial \vec{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \vec{E} \tag{4.6}$$

ここで,μ, ϵ, σ はそれぞれ透磁率, 誘電率, 導電率であり場所の関数であるとする. 電界と磁界の時間微分をそれぞれ次式のように定義すると

$$\left.\frac{\partial \vec{E}}{\partial t}\right|_{t=\left(n-\frac{1}{2}\right)\Delta t} = \frac{\vec{E}^n - \vec{E}^{n-1}}{\Delta t} \tag{4.7}$$

$$\left.\frac{\partial \vec{H}}{\partial t}\right|_{t=n\Delta t} = \frac{\vec{H}^{n+\frac{1}{2}} - \vec{H}^{n-\frac{1}{2}}}{\Delta t} \tag{4.8}$$

となり, この式を用いることで電界と磁界の式をFDTD形式に書き直すことが出来る. それは次式の通りである.

$$\frac{\vec{E}^n - \vec{E}^{n-1}}{\Delta t} = -\frac{\sigma}{\epsilon}\vec{E}^{n-\frac{1}{2}} + \frac{1}{\epsilon}\nabla \times \vec{H}^n \tag{4.9}$$

$$\frac{\vec{H}^{n+\frac{1}{2}} - \vec{H}^{n-\frac{1}{2}}}{\Delta t} = -\frac{1}{\mu}\nabla \times \vec{E}^n \tag{4.10}$$

電界 E を $n =$ 整数次の時刻に, 磁界 H を $n =$ 半奇数次の時刻に割り当てることとしているので, 時間的な平均を取り, 電界を E^n についてまとめて解くと

$$\vec{E}^n = \frac{1 - \frac{\sigma\Delta t}{2\epsilon}}{1 + \frac{\sigma\Delta t}{2\epsilon}}\vec{E}^{n-1} + \frac{\frac{\Delta t}{\epsilon}}{1 + \frac{\sigma\Delta t}{2\epsilon}}\nabla \times \vec{H}^{n-\frac{1}{2}} \tag{4.11}$$

一方磁界は $H^{n+1/2}$ についてまとめると

$$\vec{H}^{n+\frac{1}{2}} = \vec{H}^{n-1} - \frac{\Delta t}{\mu}\nabla \times \vec{E}^n \tag{4.12}$$

のようになる.FDTD の計算はこの 2 式を基本とし, これらからも分かるように FDTD 法では $t = (n-1)\Delta t$ の電界 E^{n-1} と $t = (n-1/2)\Delta t$ の磁界 $H^{n-1/2}$ から次の半ステップ後の電界 E^n が計算され, さらにこの電界 E^n と磁界 $H^{n-1/2}$ から次の半ステップ後の磁界が導出される.

中央差分を用いたことにより, 電界と磁界は空間的にも交互に配置されることとなる. 基本的には電界はセルの各辺に沿って, また磁界は面の中心に垂直に割り当てられる. これは, 電界の回転 (∇) が磁界を, 磁界の回転 (∇) が電界を作るマクスウェルの方程式を満たすようになっている. このことを表したのが図 4.2 である.

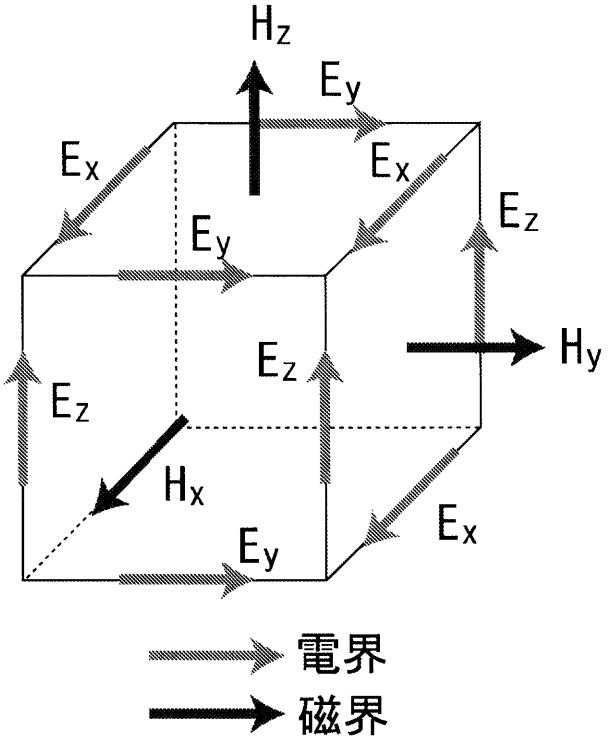

図 4.2: 単位セルにおける電磁界の空間配置

4.2.3　FDTD法の計算フローチャート

以上のことをフローチャートにまとめると図4.3になる. ただし, このフローチャートには次節で述べる吸収境界層もフローチャートに含め示してある. この部分に関しては, 次節を参考にされたい.

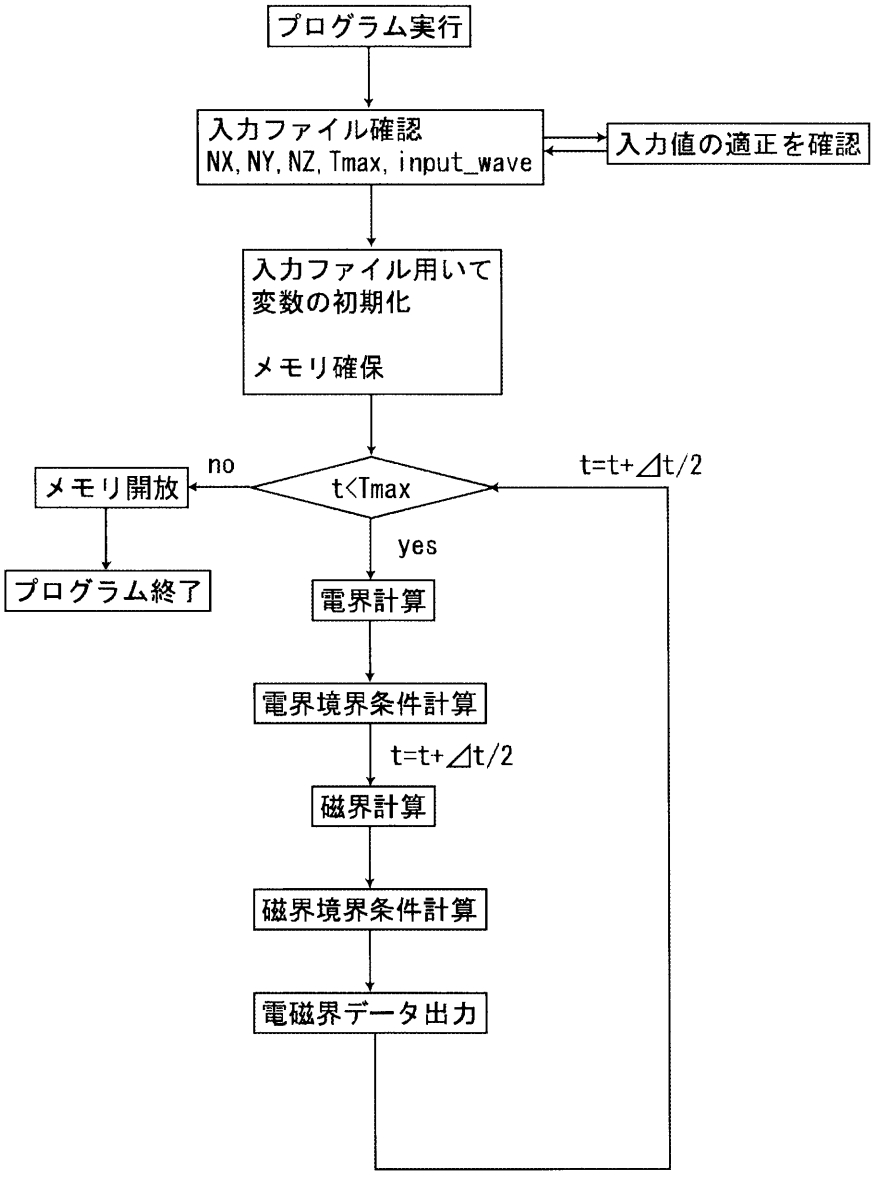

図 4.3: FDTD法の計算フローチャート

4.2.4　2次元FDTDから3次元FDTDへ

図4.2に従い,電界と磁界の座標位置を踏まえた基本セルを決定する.これを行なうには,二次元のFDTDについて考えると理解の助けになる.つまり,E_Z, H_x, H_yの成分のみ持ち,$H_Z = 0$のTM波と,H_Z, E_x, E_yの成分のみ持ち,$E_Z = 0$のTE波の二つの波をそれぞれ独立に考えるのである.これらの2次元FDTDを重ね合わせることで3次元FDTDが完成する.そこでまずは,TM波とTE波の解説を行なう.

2次元FDTD法(TM波)

解析空間をある個数の微小セルに分割する.次に各セルに電界,磁界の各成分を割り当てる.実際に3*3セルに分割した解析空間を図4.4のように用意した.プログラミングの際に必要なのものは,x,y,z方向の座標の取り方である.図中のように設定すると,ソースの記述が統一できる.式で表すと次のとおりである.

$$\left.\begin{array}{l} E_z\left(i, j, k+\frac{1}{2}\right) \Rightarrow EZ(I, J, K) \\ H_x\left(i, j+\frac{1}{2}, k+\frac{1}{2}\right) \Rightarrow HX(I, J, K) \\ H_y\left(i+\frac{1}{2}, j, k+\frac{1}{2}\right) \Rightarrow HY(I, J, K) \end{array}\right\} \tag{4.13}$$

3*3セルの場合,E_z, H_x, H_yの計算範囲は図4.4のようになる.次にこれを一般化する.NX*NYセルの計算を行う時,計算範囲は表4.1のようになると考えられる.

表4.1: TM波計算範囲

	計算始点	計算終点
Ez	1,1	NX,NY
Hx	1,0	NX,NY
Hy	0,1	NX,NY

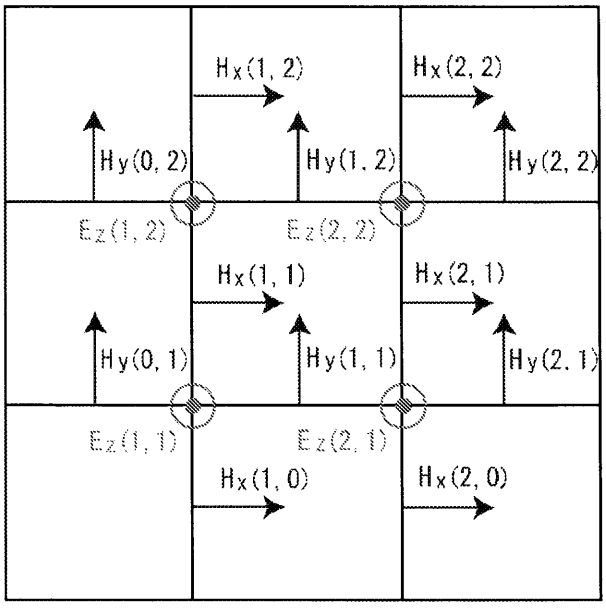

図 4.4: 2次元FDTD法TM波3*3セル

2次元FDTD法(TE波)

次に2次元FDTD法(TE波)についてまとめる. これについては上述の2次元FDTD法TM波とほぼ同様にまとめることが出来る. 図4.5には3*3セルに分割した解析空間を用意した. 座標の設定は次式の通りである.

$$\left.\begin{array}{l} E_x\left(i+\frac{1}{2},j,k\right) \Rightarrow EX(I,J,K) \\ E_y\left(i,j+\frac{1}{2},k\right) \Rightarrow EY(I,J,K) \\ H_z\left(i+\frac{1}{2},j+\frac{1}{2},k\right) \Rightarrow HZ(I,J,K) \end{array}\right\} \tag{4.14}$$

3*3セルの場合,H_z, E_x, E_yの計算範囲は図4.5のようになる. 次にこれを一般化する.NX*NYセルの計算を行う時, 計算範囲は表4.2のようになると考えられる.

表 4.2: TE波計算範囲

	計算始点	計算終点
Hz	0,0	NX,NY
Ex	0,1	NX,NY
Ey	1,0	NX,NY

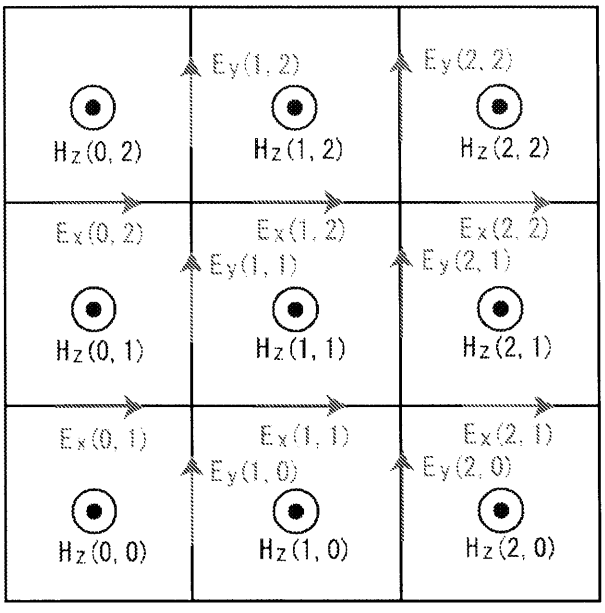

図 4.5: 2次元FDTD法TE波3*3セル

以上のことをまとめると3次元FDTDプログラムの数式においては, 座標はTE波,TM波を足し合わせた形で次式のようになる.

$$\left.\begin{aligned} E_x\left(i+\tfrac{1}{2},j,k\right) &\Rightarrow EX(I,J,K) \\ E_y\left(i,j+\tfrac{1}{2},k\right) &\Rightarrow EY(I,J,K) \\ E_z\left(i,j,k+\tfrac{1}{2}\right) &\Rightarrow EZ(I,J,K) \\ H_x\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right) &\Rightarrow HX(I,J,K) \\ H_y\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) &\Rightarrow HY(I,J,K) \\ H_z\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right) &\Rightarrow HZ(I,J,K) \end{aligned}\right\} \tag{4.15}$$

表 4.3: TM・TE合成波計算範囲

	計算始点	計算終点
Ex	0,1	NX,NY
Ey	1,0	NX,NY
Ez	1,1	NX,NY
Hx	1,0	NX,NY
Hy	0,1	NX,NY
Hz	0,0	NX,NY

ここで,3次元のプログラムを組む際に重要な設定がある.TE波,TM波の上下関係をどのように決めるかである. すでに式中では示されているが, 本研究ではTE波の上にTM波が配置された形として定義した. これを上から見て図4.6のように設定した. このことを元に3次元FDTDのプロ

グラムを行なうと

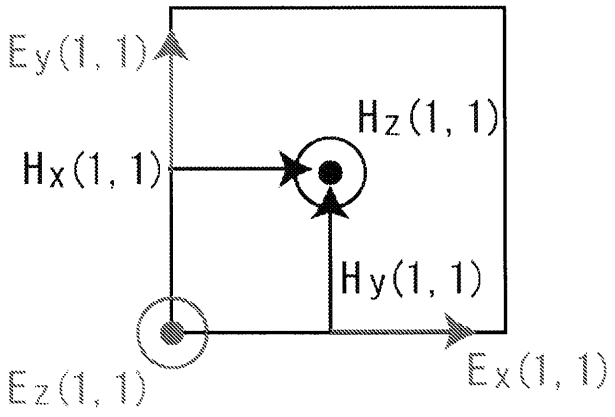

図 4.6: 基本セルの座標関係

$$
\begin{aligned}
ex[i][j][k] &= ex[i][j][k] + (dt/\sigma[i][j][k]) * (1/dy) * (hz[i][j][k] - hz[i][j-1][k]) \\
&- (dt/\sigma[i][j][k]) * (1/dz) * (hy[i][j][k] - hy[i][j][k-1]); \\
ey[i][j][k] &= ey[i][j][k] + (dt/\sigma[i][j][k]) * (1/dz) * (hx[i][j][k] - hx[i][j][k-1]) \\
&- (dt/\sigma[i][j][k]) * (1/dx) * (hz[i][j][k] - hz[i-1][j][k]); \\
ez[i][j][k] &= ez[i][j][k] + (dt/\sigma[i][j][k]) * (1/dx) * (hy[i][j][k] - hy[i-1][j][k]) \\
&- (dt/\sigma[i][j][k]) * (1/dy) * (hx[i][j][k] - hx[i][j-1][k]); \\
hx[i][j][k] &= hx[i][j][k] - (dt/\mu[i][j][k]) * (1/dy) * (ez[i][j+1][k] - ez[i][j][k]) \\
&+ (dt/\mu[i][j][k]) * (1/dz) * (ey[i][j][k+1] - ey[i][j][k]); \\
hy[i][j][k] &= hy[i][j][k] - (dt/\mu[i][j][k]) * (1/dz) * (ex[i][j][k+1] - ex[i][j][k]) \\
&+ (dt/\mu[i][j][k]) * (1/dx) * (ez[i+1][j][k] - ez[i][j][k]); \\
hz[i][j][k] &= hz[i][j][k] - (dt/\mu[i][j][k]) * (1/dx) * (ey[i+1][j][k] - ey[i][j][k]) \\
&+ (dt/\mu[i][j][k]) * (1/dy) * (ex[i][j+1][k] - ex[i][j][k]);
\end{aligned}
$$

のように表わすことが出来る. 注意すべき点は, 電界の添え字 [i][j][k] は差分化する部分で, 第二項がマイナスになり, 磁界の添え字 [i][j][k] は差分する部分は第一項がプラスになると言うことである. また繰り返しになるが,μ, σ は場所の関数であるので, 計算対象 (左辺) の座標位置と関係があ

る, このことを留意してプログラミングする必要がある.

4.2.5　セルサイズ

FDTD法は差分が基本であるから, セルサイズは細かければ細かいほど精度の高い結果を得ることが出来る. しかし, 演算時間とメモリサイズの関係から, 実際の計算においてはどの程度細かくすべきかが問題になる. これは, どのくらい正確な結果を必要としているかにかかわるが, 一般的には, 問題とする最大周波数に対してセルの一辺を1/10波長程度にする, と言うことである. ではこのとき最大周波数 f_{max} はどのように選べばよいのであろうか. これに対する明確な答えは文献などにも掲載されてはいない. 文献[1]によれば, ガウスパルスを信号源として用いた際, スペクトルの最大値(例えば直流成分)から $120dB$ 小さい周波数を f_{max} とする. この時, これは直流から f_{max} までは6桁の計算精度が補償されることになる. このことを参考にセルサイズは決定すれば良い.

$$\Delta x = \Delta y = \Delta z = \Delta = \frac{v}{f_{\max}} \frac{1}{10} \tag{4.16}$$

また誘電体内部では波長が短くなるから, それに合わせてセルサイズも細かくする必要がある. しかし大きな誘電率を有する物体と真空が混在するような状況には, 誘電体内部にだけ細かなセルを適応するのが, メモリの面で有効である. このことについては, 本プログラムではセルサイズは一定としている.

4.2.6　時間ステップ

時間ステップ Δt は,Courantの安定条件より

$$v\Delta t \leq \frac{1}{\sqrt{\left(\frac{1}{\Delta x}\right)^2 + \left(\frac{1}{\Delta y}\right)^2 + \left(\frac{1}{\Delta z}\right)^2}} \tag{4.17}$$

としなければならない. この条件は極めて厳しく, わずかでも満足しなければ計算が不安定になる. これを避ける為にCourant基準より少し小さな値を時間ステップに採用すべきである. 本研究で作成したプログラムでは一度 Δt を求め, その後 $\Delta t = \Delta t - \Delta t * \frac{1}{100}$ としている. この条件で不安定になる事は現在の使用の中では報告がなかった. なお時間ステップをもっと細かくしても計算結果は変わらないことが文献[1]では示されている.

4.3　吸収境界条件

4.3.1　概要

FDTD法は有限要素と同様に，基本的には閉領域の解析手法である. 解析空間が完全導体で囲まれているような閉領域の問題を扱う場合には, 完全導体上で電界の接線成分を0とすればよい. しかし散乱解析あるいはアンテナ解析などの, 開放空間での問題を扱う場合には, 解析領域を仮想的な境界で閉じておく必要がある. この仮想的な境界を吸収境界といい, その条件を吸収境界条件と定義する.

吸収境界が完全でないと, そこからの反射が解析空間に戻り誤差の原因となる. 吸収境界条件は一般に垂直入射の平面派に対して有効である為, 吸収境界層に入射する波のスペクトルが, 出来るだけ垂直になるように散乱体やアンテナから吸収境界を離しておく必要が生じる. しかし, これは計算機資源と計算時間の増加をまねく. そこで従来より, いくつかの吸収境界条件が提案されてきた. 大きく分類すると, 吸収層で反射がないという近似的な微分方程式 (Mur, Higdon) を用いたもの, つまり吸収境界を面として設定したものと, 吸収境界で仮想的な減衰媒質を配置し空間で減衰させるもの (Berenger), つまり吸収境界を空間として設定したものの2種類である.

本研究では, メモリの使用量は大きくなりプログラムも複雑になるが反射波を大きく減衰させ, 解析空間領域で誤差を少ないBerengerのPML(Perfectly Matched Layer) を採用し, そのプログラミングを次節でまとめた結果を報告する.

4.3.2　基本概念

本論文ではC言語によるFDTDのプログラミング手法の提示を主眼においているので,PMLに関しても基本概念と数式を述べるにとどめる.

Berengerの PML吸収境界条件は, 必要とするメモリは増えるものの, 現在のところ最も有効な吸収境界条件である.PMLの基本概念を知るには, 図4.7のように左側の真空空間から平面波がPML層に垂直入射する場合を考えると分かりやすい. 真空の波動インピーダンス Z_0, 媒質中の波動インピーダンス Z は, ぞれぞれ

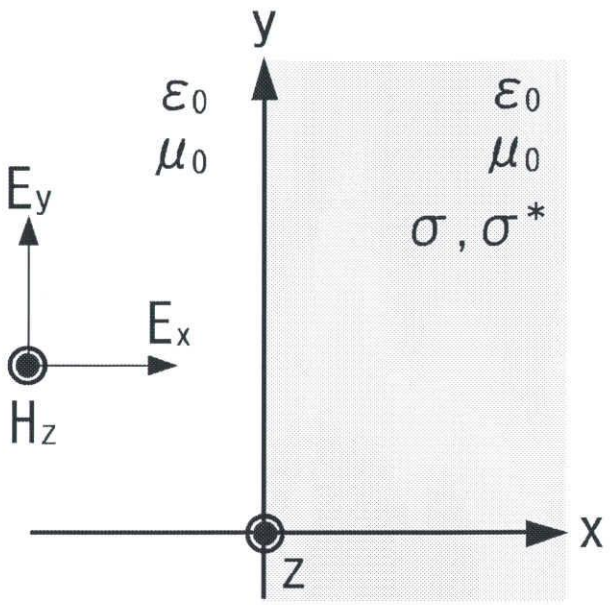

図 4.7: 磁気損失を持つ媒質への平面波の垂直入射

$$\left.\begin{aligned} Z_0 &= \sqrt{\frac{\mu_0}{\epsilon_0}} \\ Z &= \sqrt{\frac{\mu_0+\frac{\sigma^*}{j\omega}}{\epsilon_0+\frac{\sigma}{j\omega}}} = \sqrt{\frac{\mu_0\left(1+\frac{1}{j\omega}\frac{\sigma^*}{\mu_0}\right)}{\epsilon_0\left(1+\frac{1}{j\omega}\frac{\sigma}{\epsilon_0}\right)}} \end{aligned}\right\} \tag{4.18}$$

で与えられるから, インピーダンスマッチング条件,$Z_0 = Z$, すなわち

$$\frac{\sigma}{\epsilon_0} = \frac{\sigma^*}{\mu_0} \tag{4.19}$$

を満たせば, 周波数に関係なく反射係数は0になり, 電磁波は反射無しに媒質へと浸透する. したがって解析空間をこのような媒質で囲めばよいことになる. ただし σ^* は磁気導電率である. しかしこの条件を満たしても, 角度のある入射に関しては, 反射係数を完全に0にすることは不可能である.Berenger はこれを解決する為に, 新たな導電率, 導磁率を導入し, 角度のある入射に対してもマッチング条件を満たすことを考えた. それが非物理的な媒質であり,PML と呼ばれるものである. この PML 空間内では,x 方向,y 方向,z 方向それぞれ独立にインピーダンスマッチングが取れていれば良いと考える. 言い換えると,x 方向,y 方向,z 方向にも独立に平面波が進むような仮想的な媒質を考える, と言うことである. このことを式で表すと次のようになる.

$$\left.\begin{aligned} E_x &= E_{xy} + E_{xz} \\ E_y &= E_{yx} + E_{yz} \\ E_z &= E_{zx} + E_{zy} \end{aligned}\right\} \tag{4.20}$$

$$\left.\begin{aligned} H_x &= H_{xy} + H_{xz} \\ H_y &= H_{yx} + H_{yz} \\ H_z &= H_{zx} + H_{zy} \end{aligned}\right\} \tag{4.21}$$

このPML層内では,x,y,zの電磁界の成分がそれぞれ自身と垂直な2方向に分解されているので,まずそれらを計算し,次にその2成分を足し合わせることで,その場所の電磁界の成分としている.その際必要となるのは,x,y,zの電磁界の成分を格納する為の配列(メイン配列)と2つの成分をそれぞれ格納する為の配列である.次にPML内部の基本式を示す.PML内部では,基本式は12個になり

$$\left.\begin{aligned} \epsilon_0 \frac{\partial E_{xy}}{\partial t} + \sigma_y E_{xy} &= \frac{\partial H_z}{\partial y} \\ \epsilon_0 \frac{\partial E_{xz}}{\partial t} + \sigma_z E_{xz} &= -\frac{\partial H_y}{\partial z} \end{aligned}\right\} \tag{4.22}$$

$$\left.\begin{aligned} \epsilon_0 \frac{\partial E_{yz}}{\partial t} + \sigma_z E_{yz} &= \frac{\partial H_x}{\partial z} \\ \epsilon_0 \frac{\partial E_{yx}}{\partial t} + \sigma_x E_{yx} &= -\frac{\partial H_z}{\partial x} \end{aligned}\right\} \tag{4.23}$$

$$\left.\begin{aligned} \epsilon_0 \frac{\partial E_{zx}}{\partial t} + \sigma_x E_{zx} &= \frac{\partial H_y}{\partial x} \\ \epsilon_0 \frac{\partial E_{zy}}{\partial t} + \sigma_y E_{zy} &= -\frac{\partial H_x}{\partial y} \end{aligned}\right\} \tag{4.24}$$

$$\left.\begin{aligned} \mu_0 \frac{\partial H_{xy}}{\partial t} + \sigma_y^* H_{xy} &= -\frac{\partial E_z}{\partial y} \\ \mu_0 \frac{\partial H_{xz}}{\partial t} + \sigma_z^* H_{xz} &= \frac{\partial E_y}{\partial z} \end{aligned}\right\} \tag{4.25}$$

$$\left.\begin{aligned} \mu_0 \frac{\partial H_{yz}}{\partial t} + \sigma_z^* H_{yz} &= -\frac{\partial E_x}{\partial z} \\ \mu_0 \frac{\partial H_{yx}}{\partial t} + \sigma_x^* H_{yx} &= \frac{\partial E_z}{\partial x} \end{aligned}\right\} \tag{4.26}$$

$$\left.\begin{aligned} \mu_0 \frac{\partial H_{zx}}{\partial t} + \sigma_x^* H_{zx} &= -\frac{\partial E_y}{\partial x} \\ \mu_0 \frac{\partial H_{zy}}{\partial t} + \sigma_y^* H_{zy} &= \frac{\partial E_x}{\partial y} \end{aligned}\right\} \tag{4.27}$$

$$\tag{4.28}$$

となる.マッチング条件は

$$\left.\begin{aligned} \frac{\sigma_x}{\epsilon_0} &= \frac{\sigma_x^*}{\mu_0} \\ \frac{\sigma_y}{\epsilon_0} &= \frac{\sigma_y^*}{\mu_0} \\ \frac{\sigma_z}{\epsilon_0} &= \frac{\sigma_z^*}{\mu_0} \end{aligned}\right\} \tag{4.29}$$

である.ただし,xに垂直な面に対しては$\sigma_y = \sigma_z = 0$,yに垂直な面に対しては$\sigma_x = \sigma_z = 0$,そしてzに垂直な面に対しては$\sigma_x = \sigma_y = 0$である.PML内部での定式化であるが,電磁界が指数関

数的に減衰する為, その効果を取り入れた指数関数的時間ステップを用いなければならないとされている. しかし実際にはYeeアルゴリズムを用いてもほとんど結果は変わらないことが知られている. そこでPML媒質内での定式化を2例だけ提示するが, 残りの10式についても同様に計算すればよい.

$$
\begin{aligned}
E_{xy}^{n}\left(i+\frac{1}{2},j,k\right) &= \frac{1-\frac{\sigma_y(j)\ \Delta t}{2\epsilon_0}}{1+\frac{\sigma_y(j)\ \Delta t}{2\epsilon_0}}\ E_{xy}^{n-1}\left(i+\frac{1}{2},j,k\right) \\
&+ \frac{\frac{\Delta t}{\epsilon_0}}{1+\frac{\sigma_y(j)\ \Delta t}{2\epsilon_0}}\ \frac{1}{\Delta y}\left\{H_z^{n-\frac{1}{2}}\left(i+\frac{1}{2},j+\frac{1}{2},k\right)\right. \\
&- \left. H_z^{n-\frac{1}{2}}\left(i+\frac{1}{2},j-\frac{1}{2},k\right)\right\} \qquad (4.30)
\end{aligned}
$$

$$
\begin{aligned}
H_{zx}^{n+\frac{1}{2}}\left(i+\frac{1}{2},j+\frac{1}{2},k\right) &= \frac{1-\frac{\sigma_x\left(i+\frac{1}{2}\right)\ \Delta t}{2\epsilon_0}}{1+\frac{\sigma_x\left(i+\frac{1}{2}\right)\ \Delta t}{2\epsilon_0}}\ H_{zx}^{n-\frac{1}{2}}\left(i+\frac{1}{2},j+\frac{1}{2},k\right) \\
&- \frac{\frac{\Delta t}{\mu_0}}{1+\frac{\sigma_x\left(i+\frac{1}{2}\right)\ \Delta t}{2\epsilon_0}}\ \frac{1}{\Delta x}\ \left\{E_y^{n}\left(i+1,j+\frac{1}{2},k\right)\right. \\
&- \left. E_y^{n}\left(i,j+\frac{1}{2},k\right)\right\} \qquad (4.31)
\end{aligned}
$$

4.3.3　減衰項 (導電率, 磁気導電率) の設定法

最後にこれらの数式で用いられる減衰項についてまとめる. マッチング条件を満たしていれば, 任意の入射角, 任意の周波数で反射係数が0になる. しかしプログラミング上どうしてもPMLを有限の厚さで打ち切らなければならない. そうするとPMLの外壁でも幾分かの反射が起こることとなる. そこでPMLを複数層積層し, 徐々に損失を大きくすることによって, 電磁界を十分減衰させる方法がとられる. こうすることで, 外壁を完全電気壁もしくは完全磁気壁で囲んだことと等価になる. 以上をもとにPML吸収境界と各領域での導電率, 磁気導電率を示したのが図4.8であり, 数式で書くと次式のようになる.

$$
\sigma_x = \begin{cases} \sigma_{max}\left[\frac{L\Delta x - x}{L\Delta x}\right]^M : x < L\Delta x \\ 0 : L\Delta x < x < (NX - L - 1)\Delta x \\ \sigma_{max}\left[\frac{x-(NX-L-1)\Delta x}{L\Delta x}\right]^M\ x > (NX - L - 1)\Delta x \end{cases} \qquad (4.32)
$$

$$\sigma_y = \begin{cases} \sigma_{max}\left[\frac{L\Delta y - y}{L\Delta y}\right]^M : y < L\Delta y \\ 0 : L\Delta y < y < (NY - L - 1)\Delta y \\ \sigma_{max}\left[\frac{y-(NY-L-1)\Delta y}{L\Delta y}\right]^M y > (NY - L - 1)\Delta y \end{cases} \tag{4.33}$$

$$\left.\begin{array}{l} \sigma_x^* = \frac{\mu_0}{\epsilon_0}\sigma_x \\ \sigma_y^* = \frac{\mu_0}{\epsilon_0}\sigma_y \end{array}\right\} \tag{4.34}$$

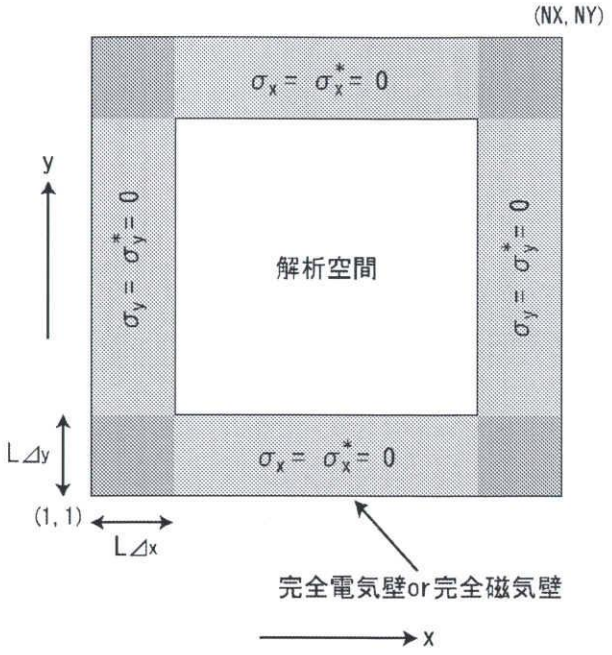

図 4.8: 基本セルの座標関係

ただし,L はPML層の層数,σ_{max} は外壁での導電率,M は導電率の分布を与える次数である. またz方向についても同様の考え方で減衰項を設定する.

4.3.4　3次元PML層

FDTDは差分を基本としているので, プログラミングする際に必要となるのは, どこからどこまで計算するか, 言い換えれば計算のループ数の設定をどのようにするか, である. 一つの方法として, 本研究ではメインとなるの配列に解析空間のセルサイズ分とPML層のセルサイズ分とを合わせたメイン配列を作り, それとは別に解析空間を囲むように26個のPML層を設けた. そしてそれぞれ番号をつけ, その番号ごとに必要な配列を用意し, メイン配列を用いて解析空間を計算し, その後減

衰に関する計算を行なう. 最後に2成分に分解したものを足し合わせてメイン配列に書き込み, その座標の数値とする方法をとっている. まず, 番号のつけ方を図4.9-4.15に示す.

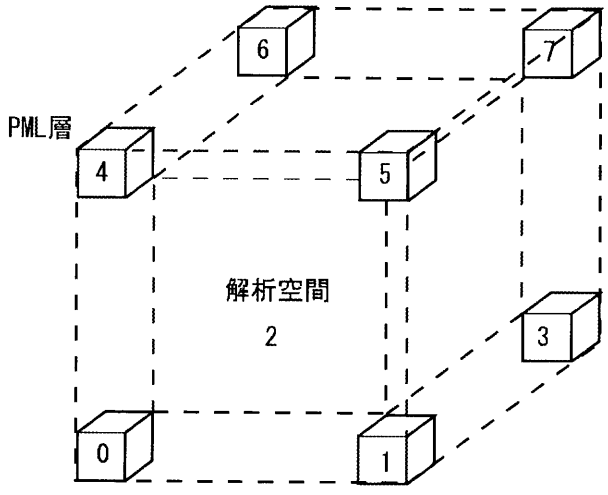

図 4.9: PML 番号 0-7

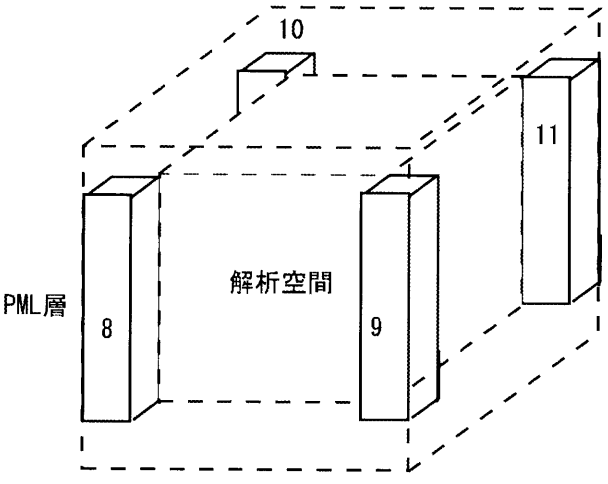

図 4.10: PML 番号 8-11

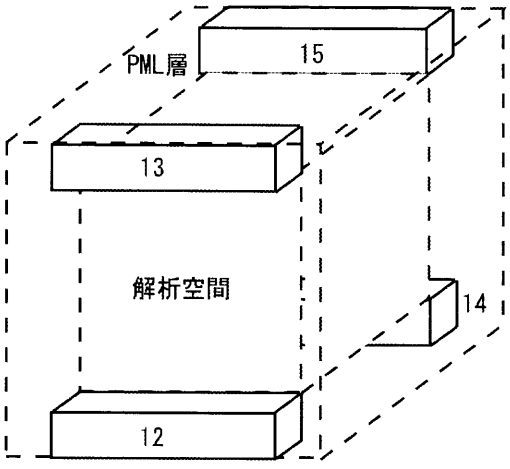

図 4.11: PML 番号 12-15

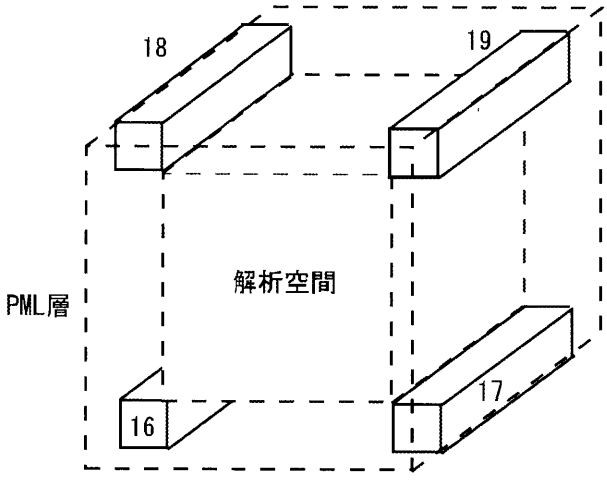

図 4.12: PML 番号 16-19

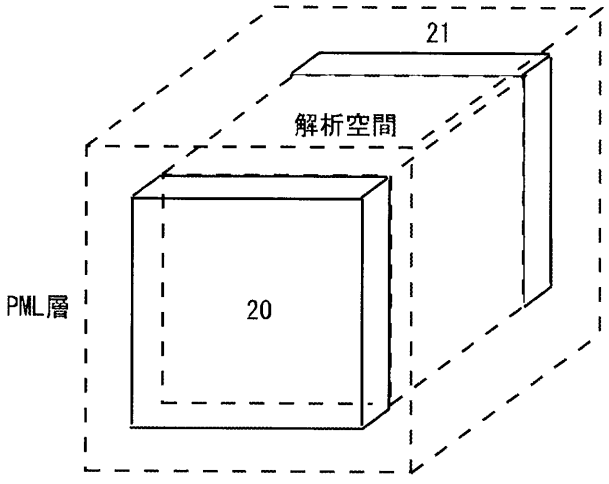

図 4.13: PML 番号 20-21

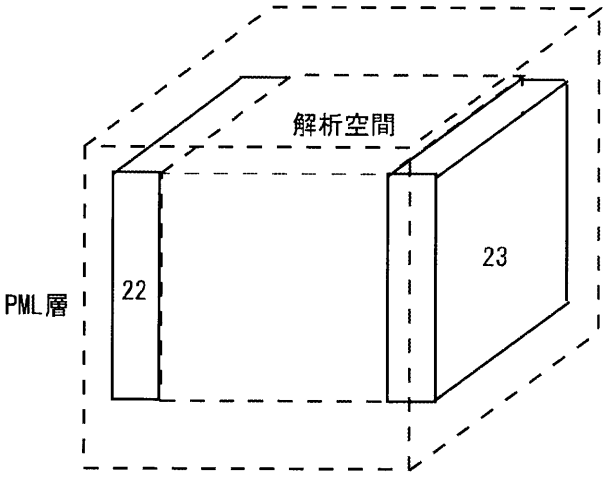

図 4.14: PML 番号 22-23

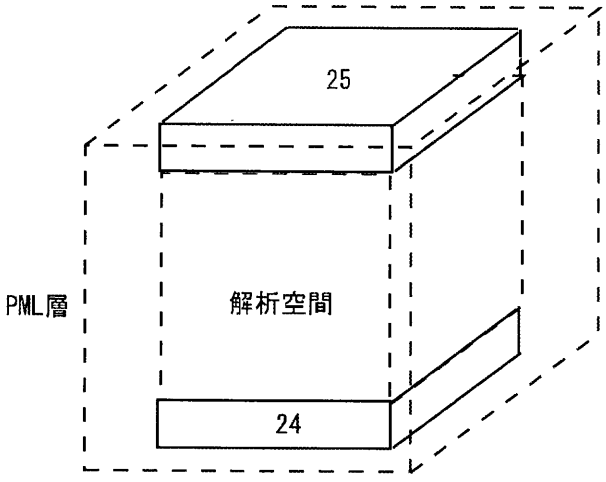

図 4.15: PML 番号 24-25

以上のように番号を割り振った配列にはサイズの定義が必要である.PML 層内の配列サイズを pml_xdim,pml_ydim,pml_zdim と設定し, 解析空間のサイズを xdim,ydim,zdim と設定すると次のようにまとめることが出来る.

```
/* ------------------------------------------------
PML のサイズをまとめ
------------------------------------------------ */

以下に PML の要素番号とそこの PML 層サイズを書き出す

number:0to7

i:pml_xdim
j:pml_ydim
k:pml_zdim

number:8to11

i:pml_xdim
j:pml_ydim
k:zdim

number:12to15

i:xdim
j:pml_ydim
k:pml_zdim

number:16to19

i:pml_xdim
j:ydim
k:pml_zdim

number20to21

i:xdim
j:pml_ydim
k:zdim
```

```
number:22to23

i:pml_xdim
j:ydim
k:zdim

numner:24to25

i:xdim
j:ydim
k:pml_zdim
```

次にループに関しては, どの配列がどこの境界面まで計算するかと言うことが問題になる. 具体的に図を参照しながら話を進める. まずは図 4.16 を用いて TM 波から説明する. 図では 2*2 の解析空間と 2*2 の PML 層を 8 個用意した. この際, プログラム上では,6*6 のメイン配列 1 つと 2*2 の PML 配列 8 つを用意する必要がある. ここで理解を助ける為に図 4.17 のように配列に番号をつける.

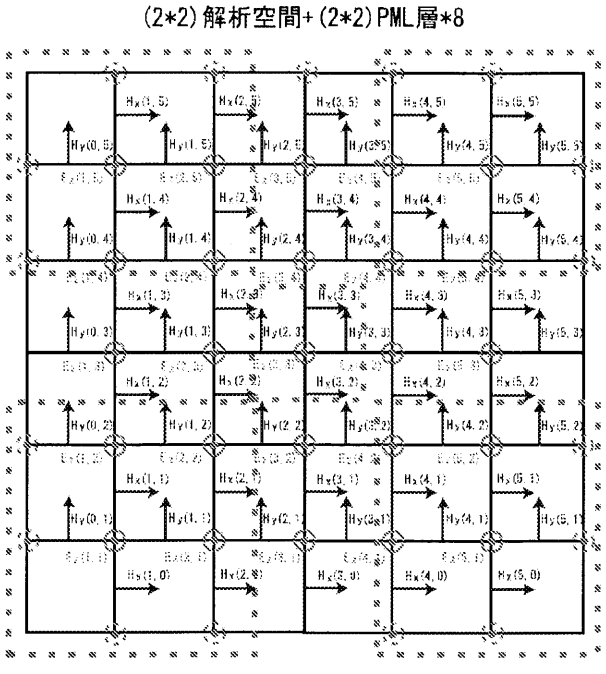

図 4.16: TM 波の計算範囲

この番号を使って図 4.16 における配列の計算範囲を説明する. まず解析空間であるが, これは前述の TM 波の計算と同じである. ただし, 座標は PML 層がメイン配列に付加されているため, その分だけ x 方向,y 方向に数値を増やす必要がある. その増加量は PML 層のセルサイズだけ, この例では,2*2 分だけ増やせばよい.

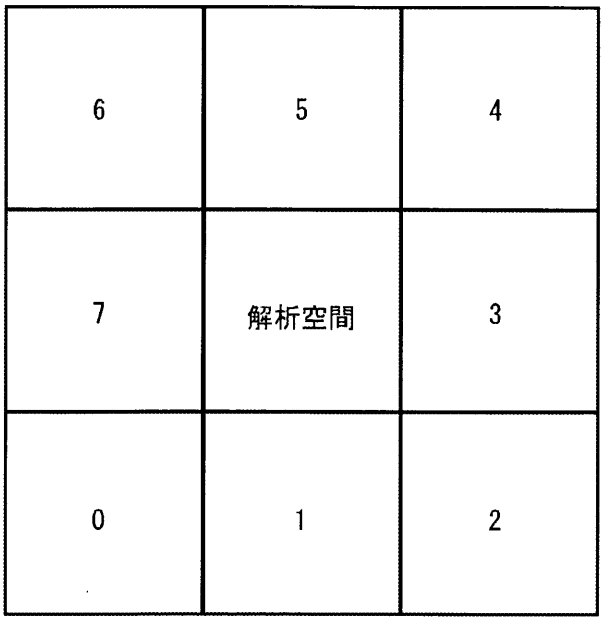

図 4.17: PML 配列の番号例

次に配列 0 に関して説明する. 配列 0 は図 4.16 中において点線で囲まれた部分を計算することとなる. 具体的には,H_x は x 方向に 1 セル分,H_y は y 方向に 1 セル分, そして E_z は x 方向と y 方向にそれぞれ 1 セル分, 通常の解析空間の計算より多く処理を行なわなければならない.

次に配列 1 に関しての説明に移る. 配列 1 は図 4.16 中において点線で囲まれていない部分を計算することとなる. 具体的には,H_y は y 方向に 1 セル分, そして E_z は y 方向に 1 セル分, 通常の解析空間の計算より多く処理を行なわなければならない.

このように全ての配列についてどこまでの計算を行なうべきかルール化するとプログラミングが楽になる. また TM 波は外壁が完全電気壁, つまり外壁部分の接線方向の電界は 0 になっている点に注意する必要がある. すなわち TM 波の場合では, 図 4.16 中の○で描かれている E_z 成分が外壁上にあるときは計算を行なわない, と言うことである.

次に TE 波である. これは図 4.18 を参照し,TM 波と同様に考えればよい. ただしこの場合も, 外壁が完全電気壁, つまり外壁部分の接線方向の電界は 0 になっている点に注意すべきである.

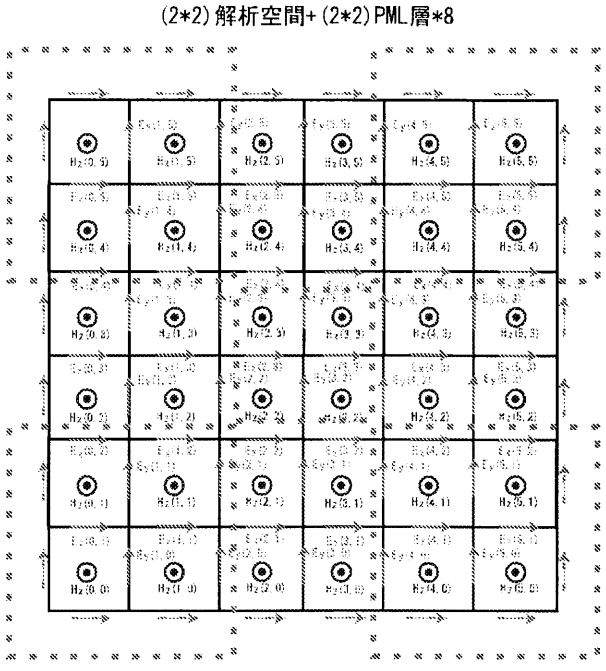

図 4.18: TE波の計算範囲

上述の議論より次のことが言える. 解析空間に接する面 (若しくは線) 方向に対して,PML層は解析空間の計算より1成分余分 (若しくはセルの一部分だけ余分) に計算する必要がある. このことを解析空間から見れば,PML層に接する境界面 (線) 上の成分の計算は,PML配列上で行なっていると言うことである.

この考え方は3次元に関しても同様である.3次元の場合はPML層同士の境界面も存在する. その事を考慮に入れ,PML番号ごとに一般化すると,PML層0〜7番は, 解析空間と点で接し通常の計算よりx,y,z全ての方向で1セル分余分に計算する必要がある.PML層8〜19番までは, 解析空間と線で接し通常の計算よりx,y,zの方向のうちいずれか二つの方向で1セル分余分に計算する必要がある.PML層20〜25番までは, 解析空間と面で接し通常の計算よりx,y,zの方向のうちいずれか一つの方向で1セル分余分に計算する必要がある. 全ての場合において余分に計算する方向は, そのPML層と解析空間の位置関係により決まる. すなわち,PML層から解析空間に向かう方向である.

最後にこれらの事柄の詳細を文章にまとめたので,プログラムの際に参考にしていただきたい.

```
/* ------------------------------------------------
PMLの式に関してまとめ
------------------------------------------------ */

●電界の添え字:
差分する部分は第二項がマイナス1

●磁界の添え字:
差分する部分は第一項がプラス1

/* ■■■■■■■ Ex ■■■■■■■ *//*●*/

exy[i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[i][j][k]
             + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(hz[i][j][k]-hz[i][j-1][k]);
exz[i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[i][j][k]
             - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][j][k]-hy[i][j][k-1]);
ex[i][j][k] = exy[i][j][k] + exz[i][j][k];

/* ■■■■■■■ Ey ■■■■■■■ *//*●*/

eyx[i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[i][j][k]
             - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][j][k]-hz[i-1][j][k]);
eyz[i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[i][j][k]
             + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hx[i][j][k]-hx[i][j][k-1]);
ey[i][j][k] = eyx[i][j][k] + eyz[i][j][k];

/* ■■■■■■■ Ez ■■■■■■■ *//*●*/

ezx[i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[i][j][k]
             + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][j][k]-hy[i-1][j][k]);
ezy[i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[i][j][k]
             - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(hx[i][j][k]-hx[i][j-1][k]);
ez[i][j][k] = ezx[i][j][k] + ezy[i][j][k];

/* ■■■■■■■ Hx ■■■■■■■ *//*●*/

hxy[i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[i][j][k]
             - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ez[i][j+1][k]-ez[i][j][k]);
hxz[i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[i][j][k]
             + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][j][k+1]-ey[i][j][k]);
hx[i][j][k] = hxy[i][j][k] + hxz[i][j][k];

/* ■■■■■■■ Hy ■■■■■■■ *//*●*/

hyx[i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[i][j][k]
             + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][j][k]-ez[i][j][k]);
hyz[i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[i][j][k]
             - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][j][k+1]-ex[i][j][k]);
hy[i][j][k] = hyx[i][j][k] + hyz[i][j][k];

/* ■■■■■■■ Hz ■■■■■■■ *//*●*/

hzx[i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[i][j][k]
             - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][j][k]-ey[i][j][k]);
hzy[i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[i][j][k]
             + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ex[i][j+1][k]-ex[i][j][k]);
hz[i][j][k] = hzx[i][j][k] + hzy[i][j][k];

/* ------------------------------------------------
電界,磁界のループ(解析空間)に関してまとめ
------------------------------------------------ */

/* ■■■■■■■ Ex ■■■■■■■ *//*●*/

for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
```

```
  }
 }
}

/* ■■■■■■■ Ey ■■■■■■■ *//*●*/

for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
  }
 }
}

/* ■■■■■■■ Ez ■■■■■■■ *//*●*/

for(i=1;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
  }
 }
}

/* ■■■■■■■ Hx ■■■■■■■ *//*●*/

for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
  }
 }
}

/* ■■■■■■■ Hy ■■■■■■■ *//*●*/

for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
  }
 }
}

/* ■■■■■■■ Hz ■■■■■■■ *//*●*/

for(i=0;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   上式;
  }
 }
}

/* ------------------------------------------------
PML のループに関してまとめ
------------------------------------------------ */

x 方向にループサイズが変化した場合
適用されるのは

Ey,Ez,Hx

y 方向にループサイズが変化した場合
適用されるのは

Ex,Ez,Hy

z 方向にループサイズが変化した場合
適用されるのは
```

```
Ex,Ey,Ez,Hx,Hy,Hz

/* ■■■■■■■ PML number 0 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"+1"
z 方向に"+1"

/* ■■■■■■■ PML number 1 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"+1"
z 方向に"+1"

/* ■■■■■■■ PML number 2 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"-1"
z 方向に"+1"

/* ■■■■■■■ PML number 3 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"-1"
z 方向に"+1"

/* ■■■■■■■ PML number 4 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"+1"
z 方向に"-1"

/* ■■■■■■■ PML number 5 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"+1"
z 方向に"-1"

/* ■■■■■■■ PML number 6 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"-1"
z 方向に"-1"

/* ■■■■■■■ PML number 7 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"-1"
z 方向に"-1"

/* ■■■■■■■ PML number 8 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"+1"

/* ■■■■■■■ PML number 9 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"+1"

/* ■■■■■■■ PML number 10 ■■■■■■■ *//*●*/

x 方向に"+1"
y 方向に"-1"

/* ■■■■■■■ PML number 11 ■■■■■■■ *//*●*/

x 方向に"-1"
y 方向に"-1"
```

```
/* ■■■■■■■ PML number 12 ■■■■■■■ *//*●*/

y 方向に"+1"
z 方向に"+1"

/* ■■■■■■■ PML number 13 ■■■■■■■ *//*●*/

y 方向に"+1"
z 方向に"-1"

/* ■■■■■■■ PML number 14 ■■■■■■■ *//*●*/

y 方向に"-1"
z 方向に"+1"

/* ■■■■■■■ PML number 15 ■■■■■■■ *//*●*/

y 方向に"-1"
z 方向に"-1"

/* ■■■■■■■ PML number 16 ■■■■■■■ *//*●*/

x 方向に"+1"
z 方向に"+1"

/* ■■■■■■■ PML number 17 ■■■■■■■ *//*●*/

x 方向に"-1"
z 方向に"+1"

/* ■■■■■■■ PML number 18 ■■■■■■■ *//*●*/

x 方向に"+1"
z 方向に"-1"

/* ■■■■■■■ PML number 19 ■■■■■■■ *//*●*/

x 方向に"-1"
z 方向に"-1"

/* ■■■■■■■ PML number 20 ■■■■■■■ *//*●*/

y 方向に"+1"

/* ■■■■■■■ PML number 21 ■■■■■■■ *//*●*/

y 方向に"-1"

/* ■■■■■■■ PML number 22 ■■■■■■■ *//*●*/

x 方向に"+1"

/* ■■■■■■■ PML number 23 ■■■■■■■ *//*●*/

x 方向に"-1"

/* ■■■■■■■ PML number 24 ■■■■■■■ *//*●*/

z 方向に"+1"

/* ■■■■■■■ PML number 25 ■■■■■■■ *//*●*/

z 方向に"-1"
```

このようにまとまられた数式 $E_x, E_y, E_z, H_x, H_y, H_z$ の中の減衰項は, 減衰項が座標の関数であるがゆえに, それぞれPML層番号によって計算ループが異なる. そのことを次にまとめた.PML番

号は図4.9-4.15の番号に従う.

```
/* ------------------------------------------------
PMLの減衰項に関してまとめ
------------------------------------------------ */

以下にPMLの要素番号とそこのPML層に存在する
減衰項を書き出す

number:0to7

rox
roy
roz

number:8to11

rox
roy

number:12to15

roy
roz

number:16to19

rox
roz

number20to21

roy

number:22to23

rox

numner:24to25

roz

/* ■■■■■■■■ PML number 0  ■■■■■■■■ */

rox[pml_xdim-i]
roy[pml_ydim-j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 1 ■■■■■■■■ */

rox[i]
roy[pml_ydim-j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 2 ■■■■■■■■ */

rox[pml_xdim-i]
roy[j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 3 ■■■■■■■■ */

rox[i]
roy[j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 4 ■■■■■■■■ */

rox[pml_xdim-i]
roy[pml_ydim-j]
```

```
roz[k]

/* ■■■■■■■■ PML number 5 ■■■■■■■■ */

rox[i]
roy[pml_ydim-j]
roz[k]

/* ■■■■■■■■ PML number 6 ■■■■■■■■ */

rox[pml_xdim-i]
roy[j]
roz[k]

/* ■■■■■■■■ PML number 7 ■■■■■■■■ */

rox[i]
roy[j]
roz[k]

/* ■■■■■■■■ PML number 8 ■■■■■■■■ */

rox[pml_xdim-i]
roy[pml_ydim-j]

/* ■■■■■■■■ PML number 9 ■■■■■■■■ */

rox[i]
roy[pml_ydim-j]

/* ■■■■■■■■ PML number 10 ■■■■■■■■ */

rox[pml_xdim-i]
roy[j]

/* ■■■■■■■■ PML number 11 ■■■■■■■■ */

rox[i]
roy[j]

/* ■■■■■■■■ PML number 12 ■■■■■■■■ */

roy[pml_ydim-j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 13 ■■■■■■■■ */

roy[pml_ydim-j]
roz[k]

/* ■■■■■■■■ PML number 14 ■■■■■■■■ */

roy[j]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 15 ■■■■■■■■ */

roy[j]
roz[k]

/* ■■■■■■■■ PML number 16 ■■■■■■■■ */

rox[pml_xdim-i]
roz[pml_zdim-k]

/* ■■■■■■■■ PML number 17 ■■■■■■■■ */

rox[i]
roz[pml_zdim-k]
```

```
/* ■■■■■■■ PML number 18 ■■■■■■■ */

rox[pml_xdim-i]
roz[k]

/* ■■■■■■■ PML number 19 ■■■■■■■ */

rox[i]
roz[k]

/* ■■■■■■■ PML number 20 ■■■■■■■ */

roy[pml_ydim-j]

/* ■■■■■■■ PML number 21 ■■■■■■■ */

roy[j]

/* ■■■■■■■ PML number 22 ■■■■■■■ */

rox[pml_xdim-i]

/* ■■■■■■■ PML number 23 ■■■■■■■ */

rox[i]

/* ■■■■■■■ PML number 24 ■■■■■■■ */

roz[pml_zdim-k]

/* ■■■■■■■ PML number 25 ■■■■■■■ */

roz[k]
```

4.4　空間設定用入力ファイル

現在作成しているプログラムを汎用のものにする為に設定ファイルを規定し, その入力によってセル数やPML層の有無などを規定している.#をコメント行とし, そこに詳細を記述できるようにしてある. 読み込みには関数 [xxscanf](詳細な動作に関しては, 付録にソースを掲載しているので, そちらを参照されたい) を使用している.

●●● param_for_FDTD_ver401.txt ●●●

```
#
# [FDTD_ver4.01.exe] のパラメータファイル
#
# ■重要：ここで入力する座標（信号源）に関して
# プログラム内部では，解析空間配列には外側に PML 配列を内包した形をとっています
# ここで入力して欲しいの値は，解析空間のみで考えた場合の値（座標）です
# しかし配列は [0] から始まるため分かりにくい
# ↓
# ○（解析空間のみの）セル番号で入力をお願いします
#
# ■入力信号関係
#
# (0) 入力したい信号数 -- 入力した分だけ (1)～(4) に追加記述
#     書式 : "%d"
#
1 #<=★
#
```

```
# (1) 周波数
#     書式 : "%lf"
#
2400000000 #<=★
#
# 入力したい信号数が二つ以上ならば次のように追加する (現在はコメントアウト)
#
# (1') 周波数
#     書式 : "%lf"
#
#2500000000 #<=★
#
# (2) 信号源位置 (x,y,z)
#     書式 : "%d %d %d"
#
10 10 10
#
# (3) 励振時間 (start_time～end_time[s])
#     書式 : "%lf %lf"
#
0.0 0.001 #<=★
#
# (4) 計算時間 (入力信号の何周期分か)
#     書式 : "%lf"
#
40.0 #<=★
#
# ■セルサイズ関係
#
# (5) 解析空間の層数 (x, y, z)
#     書式 : "%d %d %d"
#
30 30 30 #<=★
#
# (6) 解析空間のセルサイズ (dx, dy, dz)
#     書式 : "%lf %lf %lf"
#
0.0125 0.0125 0.0125 #<=★
#
# ■境界条件関係
#
#
# (7) 境界条件の種類
#     なし    : 0
#     PML     : 1
#     Mur1 次 :
#     書式 : "%d"
#
1 #<=★
#
# (8) PML 用設定データ -- PML 層の層数 (x, y, z)
#     書式 : "%d %d %d"
#
4 4 4 #<=★
#
# (9) PML 用設定データ -- PML 層の減衰項 (M(2～3 を入れるのが鉄則)):M_x,M_y,M_z
#     書式 : "%d %d %d"
#
3 3 3 #<=★
#
# (10) PML 用設定データ -- PML 層の減衰項 (dB(最大減衰値)):dB_x,dB_y,dB_z
#     書式 : "%d %d %d"
#
120 120 120 #<=★
#
# ■出力関係
#
# (11) 計算データ出力ファイル名
#     書式 : "%s"
#
```

```
output_data.txt #<=★
#
# (12) PPM データ出力の有無
#      なし    : 0
#      あり    : 1
#      書式 : "%d"
#
1 #<=★
#
# (13) 観測点 1 点でのデータ収集の有無
#      なし    : 0
#      あり    : 1
#      書式 : "%d"
#
1 #<=★
#
# (14) 観測座標 (観測点 1 点でのデータ収集の有りの時)
#      書式 : "%d %d %d"
#
10 10 10 #<=★
#
#

●●● input_from_opengl_ver4.txt ●●●

#
# [FDTD_ver4.00.exe] のパラメータファイル (OPENGL 用)
#
# ■重要：ここで入力する座標 (信号源) に関して
# プログラム内部では，解析空間配列には外側に PML 配列を内包した形をとっています
# ここで入力して欲しいの値は，解析空間のみで考えた場合の値 (座標) です
# しかし配列は [0] から始まるため分かりにくい
# ↓
# ○ (解析空間のみの) セル番号で入力をお願いします
#
# ■材質 ID 関係
#
# (0) 材料 ID の個数
#      書式 : "%d"
#
1 #<=★
#
# (1) ID[1] の場所 (i_start, i_end, j_start, j_end, k_start, k_end)
#      書式 : "%d %d %d %d %d %d"
#
1 1 1 1 1 1 #<=★
#
# 注) 材料が多数ある場合は,(1) をその個数分繰り返し，次に (2) を繰り返しと言う形をとる
#
# (1) ID[1] の場所 (i_start, i_end, j_start, j_end, k_start, k_end)
#      書式 : "%d %d %d %d %d %d"
#
#0 29 69 99 0 99 #<=★ これは例題を示すためなのでコメントアウト
#
# (2) ID[1] のデータ (比誘電率，比透磁率，比導電率)
#      書式 : "%lf %lf %lf"
#
1.0 1.0 1.0 #<=★
# ・同材料でも存在場所の違いで別材料として扱う点
```

4.5　付録

実際にC言語を用いて,吸収境界層PML層を付加したFDTDを次に載せる.このプログラムは,文献[1]に則って作成されており,3次元で計算を行なうことが出来,かつ空間サイズを可変として設定ファイルから読み込ませることが出来るプログラムである.

●●● FDTD_ver4.00.cpp ●●●

```
#include<math.h>
#include<stdio.h>
#include<process.h>
#include<stdlib.h>
#include"set_ID.h"
#include"malloc_free.h"
#include"input_data.h"
#include"set_ID.h"
#include"debug.h"
#include"clc_time.h"

#include <string.h>
#include"..\make_ppm_P6_H191230\make_ppm.h"

/* ■■■■■■ main function ■■■■■■ */

int main(int argc, char *argv[]){

 int i, j, k, l;
 int count=0, fcount=0;
 int input_flg;
 char fname[256];
 double *rox, *roy, *roz, romax_x, romax_y, romax_z;
 double t;
 FILE *input_file_fp, *input_opengl_fp, *output_fp, *output_observe, *log_fp;
 double eps_PML, mu_PML, ro_PML;
 int ***ID;
 int check_input_flg;
 int ppm_flg, observe_flg;
 int x_obse, y_obse, z_obse;

 double tmax;
 double *f;
 double dx,dy,dz,dt,e,m,r;
 int xdim, ydim, zdim;
 int pml_xdim, pml_ydim, pml_zdim;
 int M_x, M_y, M_z, dB_x, dB_y, dB_z;
 int *f_x, *f_y, *f_z, f_num;


 double ***ex, ***ey, ***ez;
 double ****exy, ****exz, ****eyx, ****eyz, ****ezx, ****ezy;
 double ***hx, ***hy, ***hz;
 double ****hxy, ****hxz, ****hyx, ****hyz, ****hzx, ****hzy;

 char *time_data;

 /* 入力データ用 */
 input_data *input_file_data;

 /* 入力データ用 (ID配列セット用) */
 ID_data_for_setting *ID_data_for_setting_data;

 /* 入力データ用 (IDデータ用) */
 ID_data ID_data_struct;
```

```
/* コマンドライン引数に関しての定義 */
if(argc == 1){/* コマンド引数なし */
 /* 入力データファイルオープン */
 input_file_fp = fopen("param_for_FDTD_ver301.txt", "r");
 if(input_file_fp == NULL){
  printf("input_file open error\n");
  return(ERROR);
 }
 /* 入力データファイルオープン */
 input_opengl_fp = fopen("input_from_opengl_ver3.txt", "r");
 if(input_opengl_fp == NULL){
  printf("input_opengl_fp open error\n");
  return(ERROR);
 }
}else if(argc == 3){/* コマンド引数あり */
 /* 入力データファイルオープン */
 input_file_fp = fopen(argv[1], "r");
 if(input_file_fp == NULL){
  printf("input_file open error\n");
  return(ERROR);
 }
 /* 入力データファイルオープン */
 input_opengl_fp = fopen(argv[2], "r");
 if(input_opengl_fp == NULL){
  printf("input_opengl_fp open error\n");
  return(ERROR);
 }
}else{/* コマンド引数あり -- 不正 */
 printf("コマンドライン引数の個数が異なります\n");
 printf("仕様：本プログラム 入力ファイル 1 入力ファイル 2\n");
 return(ERROR);
}

/* 計算ログ書き出し用ファイルポインタ */
log_fp = fopen("log.txt", "w");
if(log_fp == NULL){
 printf("log_fp can't open file\n");
 return(ERROR);
}

/* 入力データ用メモリ確保 */
input_file_data = (input_data*)malloc(sizeof(input_data));
if(input_file_data == NULL){
 printf("input_file_data malloc error\n");
 return(ERROR);
}
ID_data_for_setting_data = (ID_data_for_setting*)malloc(sizeof(ID_data_for_setting));
if(ID_data_for_setting_data == NULL){
 printf("ID_data_for_setting_data malloc error\n");
 return(ERROR);
}

/* データ入力スタート */
input_flg = input_data_from_file(input_file_fp, input_file_data);
if(input_flg == ERROR){
 printf("input data from file error\n");
 return(ERROR);
}
input_flg = set_ID_data(input_opengl_fp, ID_data_for_setting_data, &ID_data_struct);
if(input_flg == ERROR){
 printf("input data from opengl error\n");
 return(ERROR);
}

/* 入力データの妥当性を確認 */
check_input_flg = check_input(input_file_data, ID_data_for_setting_data);
if(check_input_flg == ERROR){
 printf("入力データに誤りがあります\n");
 return(ERROR);
}
```

```
/* 材料配列の初期化 */
ID = set_ID(input_file_data, ID_data_for_setting_data);

/* 入力データの代入 */
tmax = input_file_data->tmax;
dx = input_file_data->dx;
dy = input_file_data->dy;
dz = input_file_data->dz;
dt = input_file_data->dt;
t = input_file_data->t;
e = ID_data_struct.eps[0];
m = ID_data_struct.mu[0];
r = ID_data_struct.ro[0];
xdim = input_file_data->xdim;
ydim = input_file_data->ydim;
zdim = input_file_data->zdim;
pml_xdim = input_file_data->pml_xdim;
pml_ydim = input_file_data->pml_ydim;
pml_zdim = input_file_data->pml_zdim;
M_x = input_file_data->M_x;
M_y = input_file_data->M_y;
M_z = input_file_data->M_z;
dB_x = input_file_data->dB_x;
dB_y = input_file_data->dB_y;
dB_z = input_file_data->dB_z;
f_num = input_file_data->f_num;
ppm_flg = input_file_data->ppm_flg;
observe_flg = input_file_data->observe_flg;
x_obse = input_file_data->x_obse-1;
y_obse = input_file_data->y_obse-1;
z_obse = input_file_data->z_obse-1;

/* 観測点でのデータ収集 */
if(observe_flg == 1){
 /* ファイルオープン */
 output_observe = fopen("output_observe.txt", "w");
 if(output_observe == NULL){
  printf("output_observe file open error\n");
  return(ERROR);
 }
}


/* 信号（周波数と座標）に関して malloc */
/* 開放を忘れずに */
f = (double*)malloc(sizeof(double)*(size_t)input_file_data->f_num);
if(f == NULL){
 printf("f malloc error\n");
 return(ERROR);
}
f_x = (int*)malloc(sizeof(int)*(size_t)input_file_data->f_num);
if(f_x == NULL){
 printf("f_x malloc error\n");
 return(ERROR);
}
f_y = (int*)malloc(sizeof(int)*(size_t)input_file_data->f_num);
if(f_y == NULL){
 printf("f_y malloc error\n");
 return(ERROR);
}
f_z = (int*)malloc(sizeof(int)*(size_t)input_file_data->f_num);
if(f_z == NULL){
 printf("f_z malloc error\n");
 return(ERROR);
}

/* 信号（周波数）入力データの代入 */
for(i=0;i<input_file_data->f_num;i++){
 f[i] = input_file_data->f[i];
```

```
}

/* 信号（座標）入力データの代入 */
for(i=0;i<input_file_data->f_num;i++){
 f_x[i] = input_file_data->f_x[i] - 1; /* [-1] の理由：入力ファイルは座標を 1 から（ユーザーの立場では）考えるが，
実際の配列内は 0 からなので．下記も同様 */
 f_y[i] = input_file_data->f_y[i] - 1;
 f_z[i] = input_file_data->f_z[i] - 1;
}

/* 入力値の書き出し（log.txt） */
fprintf(log_fp, "tmax = %16.15lf\n", tmax);
fprintf(log_fp, "dx = %lf\n", dx);
fprintf(log_fp, "dy = %lf\n", dy);
fprintf(log_fp, "dz = %lf\n", dz);
fprintf(log_fp, "t = %16.15lf\n", t);
fprintf(log_fp, "e = %16.15lf\n", e);
fprintf(log_fp, "m = %16.15lf\n", m);
fprintf(log_fp, "r = %16.15lf\n", r);
fprintf(log_fp, "xdim = %d\n", xdim);
fprintf(log_fp, "ydim = %d\n", ydim);
fprintf(log_fp, "zdim = %d\n", zdim);
fprintf(log_fp, "pml_xdim = %d\n", pml_xdim);
fprintf(log_fp, "pml_ydim = %d\n", pml_ydim);
fprintf(log_fp, "pml_zdim = %d\n", pml_zdim);
fprintf(log_fp, "M_x = %d M_y = %d M_z = %d\n", M_x, M_y, M_z);
fprintf(log_fp, "dB_x = %d dB_y = %d dB_z = %d\n", dB_x, dB_y, dB_z);
fprintf(log_fp, "f_num = %d\n", f_num);
fprintf(log_fp, "ppm_flg = %d\n", ppm_flg);
fprintf(log_fp, "observe_flg = %d\n", observe_flg);
if(observe_flg == 1){
 fprintf(log_fp, "x_obse = %d y_obse = %d z_obse = %d\n", x_obse, y_obse, z_obse);
}

for(i=0;i<input_file_data->f_num;i++){
 fprintf(log_fp, "f[%d] = %lf\n", i, f[i]);
 fprintf(log_fp, "f_x[%d] = %d\n", i, f_x[i]);
 fprintf(log_fp, "f_y[%d] = %d\n", i, f_y[i]);
 fprintf(log_fp, "f_z[%d] = %d\n", i, f_z[i]);
}

/* 計算所要時間計測開始点 */
start_clc_time();

/* ------------------------------------------------------------------------------------------
ここから，計算用配列確保を行います
これも開放がだいぶ下にありますので，気をつけてください
また変数の後ろにある数字はPML層の番号に当たります
番号は次のファイルを参照
『PML_name2.ai』
------------------------------------------------------------------------------------------ */

ex = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);
ey = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);
ez = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);
hx = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);
hy = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);
hz = malloc_3Darray(xdim + 2*pml_xdim, ydim + 2*pml_ydim, zdim + 2*pml_zdim);

exy = malloc_3Darray_PML(input_file_data);
exz = malloc_3Darray_PML(input_file_data);
eyx = malloc_3Darray_PML(input_file_data);
eyz = malloc_3Darray_PML(input_file_data);
ezx = malloc_3Darray_PML(input_file_data);
ezy = malloc_3Darray_PML(input_file_data);
hxy = malloc_3Darray_PML(input_file_data);
hxz = malloc_3Darray_PML(input_file_data);
hyx = malloc_3Darray_PML(input_file_data);
hyz = malloc_3Darray_PML(input_file_data);
hzx = malloc_3Darray_PML(input_file_data);
```

```
hzy = malloc_3Darray_PML(input_file_data);

/* ------------------------------------------------------------------------------------------------
ここから，入力データ取り込みよう配列の開放
------------------------------------------------------------------------------------------------ */

free(input_file_data->f_x);
free(input_file_data->f_y);
free(input_file_data->f_z);
free(input_file_data->f);
free(input_file_data->f_start);
free(input_file_data->f_end);
free(input_file_data);

free(ID_data_for_setting_data->ID_i_start);
free(ID_data_for_setting_data->ID_i_end);
free(ID_data_for_setting_data->ID_j_start);
free(ID_data_for_setting_data->ID_j_end);
free(ID_data_for_setting_data->ID_k_start);
free(ID_data_for_setting_data->ID_k_end);
free(ID_data_for_setting_data);

/* ------------------------------------------------------------------------------------------------
ここから，減衰項の配列確保，数値決定を行い
開放がだいぶ下にありますので，気をつけてください
------------------------------------------------------------------------------------------------ */

rox = (double *)malloc(sizeof(double)*(size_t)(input_file_data->pml_xdim));
if(rox == NULL){
 free(rox);
 printf("rox malloc error!\n");
}

roy = (double *)malloc(sizeof(double)*(size_t)(input_file_data->pml_ydim));
if(roy == NULL){
 free(roy);
 printf("roy malloc error!\n");
}

roz = (double *)malloc(sizeof(double)*(size_t)(input_file_data->pml_zdim));
if(roz == NULL){
 free(roz);
 printf("roz malloc error!\n");
}

romax_x = (double)(M_x+1)*e*3.0*pow(10.0, 8.0)*log(dB_x)/(2.0*(double)pml_xdim*dx);
romax_y = (double)(M_y+1)*e*3.0*pow(10.0, 8.0)*log(dB_y)/(2.0*(double)pml_ydim*dy);
romax_z = (double)(M_z+1)*e*3.0*pow(10.0, 8.0)*log(dB_z)/(2.0*(double)pml_zdim*dz);

for(i=0;i<pml_xdim;i++){
 rox[i] = romax_x * pow(((double)(i+1)/(pml_xdim)),double(M_x));
}
for(i=0;i<pml_ydim;i++){
 roy[i] = romax_y * pow(((double)(i+1)/(pml_ydim)),double(M_y));
}
for(i=0;i<pml_zdim;i++){
 roz[i] = romax_z * pow(((double)(i+1)/(pml_zdim)),double(M_z));
}

/* 減衰項 romax のデータ確認 */
fprintf(log_fp, "romax_x = %16.15lf\n", romax_x);
fprintf(log_fp, "romax_y = %16.15lf\n", romax_y);
fprintf(log_fp, "romax_z = %16.15lf\n", romax_z);
for(i=0;i<pml_xdim;i++){
 fprintf(log_fp, "rox[%d] = %16.15lf\n", i, rox[i]);
}
for(i=0;i<pml_ydim;i++){
 fprintf(log_fp, "roy[%d] = %16.15lf\n", i, roy[i]);
}
for(i=0;i<pml_zdim;i++){
```

```
 fprintf(log_fp, "roz[%d] = %16.15lf\n", i, roz[i]);
}

/* ------------------------------------------------------------------------------------------
main loop
------------------------------------------------------------------------------------------ */

for(t=0.0;t<tmax;t+=(dt/2.0)){
 /* calculate Ex *//*●*/
 for(i=pml_xdim;i<pml_xdim+xdim;i++){
  for(j=pml_ydim+1;j<pml_ydim+ydim;j++){
   for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
    ex[i][j][k] = ex[i][j][k]
    + (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dy)*(hz[i][j][k]-hz[i][j-1][k])
    - (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dz)*(hy[i][j][k]-hy[i][j][k-1]);
   }
  }
 }

 /* calculate Ey *//*●*/
 for(i=pml_xdim+1;i<pml_xdim+xdim;i++){
  for(j=pml_ydim;j<pml_ydim+ydim;j++){
   for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
    ey[i][j][k] = ey[i][j][k]
    + (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dz)*(hx[i][j][k]-hx[i][j][k-1])
    - (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dx)*(hz[i][j][k]-hz[i-1][j][k]);
   }
  }
 }

 /* calculate Ez *//*●*/
 for(i=pml_xdim+1;i<pml_xdim+xdim;i++){
  for(j=pml_ydim+1;j<pml_ydim+ydim;j++){
   for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
    ez[i][j][k] = ez[i][j][k]
    + (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dx)*(hy[i][j][k]-hy[i-1][j][k])
    - (dt/ID_data_struct.eps[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dy)*(hx[i][j][k]-hx[i][j-1][k]);
   }
  }
 }

 /* ■■■■■■■■■■■■■■■■ PML for E ■■■■■■■■■■■■■■■■ */

 /* ■■■■ Ex ■■■■ */

 /* calculate PML Ex0 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=1;j<=pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    exy[0][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[0][i][j][k]
    + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[i][j][k]-hz[i][j-1][k]);
    exz[0][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[0][i][j][k]
    - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][j][k]-hy[i][j][k-1]);
    ex[i][j][k] = exy[0][i][j][k] + exz[0][i][j][k];
   }
  }
 }

 /* calculate PML Ex1 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=1;j<=pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    exy[1][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[1][i][j][k]
    + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[pml_xdim+xdim+i][j][k]-hz[pml_xdim+xdim+i][j-1][k]);
    exz[1][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[1][i][j][k]
    - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[pml_xdim+xdim+i][j][k]-hy[pml_xdim+xdim+i][j][k-1]);
    ex[pml_xdim+xdim+i][j][k] = exy[1][i][j][k] + exz[1][i][j][k];
   }
  }
 }
```

```
/* calculate PML Ex2 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[2][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[2][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(hz[i][pml_ydim+ydim+j][k]-hz[i][pml_ydim+ydim+j-1][k]);
   exz[2][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[2][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][pml_ydim+ydim+j][k]-hy[i][pml_ydim+ydim+j][k-1]);
   ex[i][pml_ydim+ydim+j][k] = exy[2][i][j][k] + exz[2][i][j][k];
  }
 }
}

/* calculate PML Ex3 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[3][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[3][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hz[pml_xdim+xdim+i][pml_ydim+ydim+j-1][k]);
   exz[3][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[3][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hy[pml_xdim+xdim+i][pml_ydim+ydim+j][k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = exy[3][i][j][k] + exz[3][i][j][k];
  }
 }
}

/* calculate PML Ex4 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[4][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[4][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[i][j][pml_zdim+zdim+k]-hz[i][j-1][pml_zdim+zdim+k]);
   exz[4][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[4][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][j][pml_zdim+zdim+k]-hy[i][j][pml_zdim+zdim+k-1]);
   ex[i][j][pml_zdim+zdim+k] = exy[4][i][j][k] + exz[4][i][j][k];
  }
 }
}

/* calculate PML Ex5 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[5][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[5][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i][j-1][pml_zdim+zdim+k]);
   exz[5][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[5][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i][j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = exy[5][i][j][k] + exz[5][i][j][k];
  }
 }
}

/* calculate PML Ex6 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[6][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[6][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
   exz[6][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[6][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ex[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = exy[6][i][j][k] + exz[6][i][j][k];
  }
 }
```

```
}

/* calculate PML Ex7 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[7][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[7][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
   exz[7][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[7][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = exy[7][i][j][k] + exz[7][i][j][k];
  }
 }
}

/* calculate PML Ex8 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[8][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[8][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[i][j][pml_zdim+k]-hz[i][j-1][pml_zdim+k]);
   exz[8][i][j][k] = exz[8][i][j][k] - (dt/e)*(1.0/dz)*(hy[i][j][pml_zdim+k]-hy[i][j][pml_zdim+k-1]);
   ex[i][j][pml_zdim+k] = exy[8][i][j][k] + exz[8][i][j][k];
  }
 }
}

/* calculate PML Ex9 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[9][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[9][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+xdim+i][j][pml_zdim+k]-hz[pml_xdim+xdim+i][j-1][pml_zdim+k]);
   exz[9][i][j][k] = exz[9][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[pml_xdim+xdim+i][j][pml_zdim+k]-hy[pml_xdim+xdim+i][j][pml_zdim+k-1]);
   ex[pml_xdim+xdim+i][j][pml_zdim+k] = exy[9][i][j][k] + exz[9][i][j][k];
  }
 }
}

/* calculate PML Ex10 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[10][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[10][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[i][pml_ydim+ydim+j][pml_zdim+k]-hz[i][pml_ydim+ydim+j-1][pml_zdim+k]);
   exz[10][i][j][k] = exz[10][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[i][pml_ydim+ydim+j][pml_zdim+k]-hy[i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ex[i][pml_ydim+ydim+j][pml_zdim+k] = exy[10][i][j][k] + exz[10][i][j][k];
  }
 }
}

/* calculate PML Ex11 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[11][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[11][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hz[pml_xdim+xdim+i][pml_ydim+ydim+j-1][pml_zdim+k]);
   exz[11][i][j][k] = exz[11][i][j][k]- (dt/e)*(1.0/dz)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = exy[11][i][j][k] + exz[11][i][j][k];
  }
 }
}
```

```
/* calculate PML Ex12 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[12][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[12][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[pml_xdim+i][j][k]-hz[pml_xdim+i][j-1][k]);
   exz[12][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[12][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[pml_xdim+i][j][k]-hy[pml_xdim+i][j][k-1]);
   ex[pml_xdim+i][j][k] = exy[12][i][j][k] + exz[12][i][j][k];
  }
 }
}

/* calculate PML Ex13 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[13][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[13][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+i][j][pml_zdim+zdim+k]-hz[pml_xdim+i][j-1][pml_zdim+zdim+k]);
   exz[13][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[13][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+i][j][pml_zdim+zdim+k]-hy[pml_xdim+i][j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+i][j][pml_zdim+zdim+k] = exy[13][i][j][k] + exz[13][i][j][k];
  }
 }
}

/* calculate PML Ex14 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[14][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[14][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+i][pml_ydim+ydim+j][k]-hz[pml_xdim+i][pml_ydim+ydim+j-1][k]);
   exz[14][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[14][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+i][pml_ydim+ydim+j][k]-hy[pml_xdim+i][pml_ydim+ydim+j][k-1]);
   ex[pml_xdim+i][pml_ydim+ydim+j][k] = exy[14][i][j][k] + exz[14][i][j][k];
  }
 }
}

/* calculate PML Ex15 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[15][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[15][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
   exz[15][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[15][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = exy[15][i][j][k] + exz[15][i][j][k];
  }
 }
}

/* calculate PML Ex16 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[16][i][j][k] = exy[16][i][j][k] + (dt/e)*(1.0/dy)*(hz[i][pml_ydim+j][k]-hz[i][pml_ydim+j-1][k]);
   exz[16][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[16][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][pml_ydim+j][k]-hy[i][pml_ydim+j][k-1]);
   ex[i][pml_ydim+j][k] = exy[16][i][j][k] + exz[16][i][j][k];
  }
 }
}
```

```
/* calculate PML Ex17 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[17][i][j][k] = exy[17][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[pml_xdim+xdim+i][pml_ydim+j][k]-hz[pml_xdim+xdim+i][pml_ydim+j-1][k]);
   exz[17][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[17][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+xdim+i][pml_ydim+j][k]-hy[pml_xdim+xdim+i][pml_ydim+j][k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+j][k] = exy[17][i][j][k] + exz[17][i][j][k];
  }
 }
}

/* calculate PML Ex18 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[18][i][j][k] = exy[18][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[i][pml_ydim+j][pml_zdim+zdim+k]-hz[i][pml_ydim+j-1][pml_zdim+zdim+k]);
   exz[18][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[18][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hy[i][pml_ydim+j][pml_zdim+zdim+k]-hy[i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ex[i][pml_ydim+j][pml_zdim+zdim+k] = exy[18][i][j][k] + exz[18][i][j][k];
  }
 }
}

/* calculate PML Ex19 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[19][i][j][k] = exy[19][i][j][k] + (dt/e)*(1.0/dy)
   *(hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i][pml_ydim+j-1][pml_zdim+zdim+k]);
   exz[19][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[19][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = exy[19][i][j][k] + exz[19][i][j][k];
  }
 }
}

/* calculate PML Ex20 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[20][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*exy[20][i][j][k]
   + (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hz[pml_xdim+i][j][pml_zdim+k]-hz[pml_xdim+i][j-1][pml_zdim+k])
   exz[20][i][j][k] = exz[20][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[pml_xdim+i][j][pml_zdim+k]-hy[pml_xdim+i][j][pml_zdim+k-1]);
   ex[pml_xdim+i][j][pml_zdim+k] = exy[20][i][j][k] + exz[20][i][j][k];
  }
 }
}

/* calculate PML Ex21 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   exy[21][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*exy[21][i][j][k]
   + (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hz[pml_xdim+i][pml_ydim+ydim+j-1][pml_zdim+k]);
   exz[21][i][j][k] = exz[21][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = exy[21][i][j][k] + exz[21][i][j][k];
  }
 }
}

/* calculate PML Ex22 *//*●*/
```

```
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   exy[22][i][j][k] = exy[22][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[i][pml_ydim+j][pml_zdim+k]-hz[i][pml_ydim+j-1][pml_zdim+k]);
   exz[22][i][j][k] = exz[22][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[i][pml_ydim+j][pml_zdim+k]-hy[i][pml_ydim+j][pml_zdim+k-1]);
   ex[i][pml_ydim+j][pml_zdim+k] = exy[22][i][j][k] + exz[22][i][j][k];
  }
 }
}

/* calculate PML Ex23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   exy[23][i][j][k] = exy[23][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hz[pml_xdim+xdim+i][pml_ydim+j-1][pml_zdim+k]);
   exz[23][i][j][k] = exz[23][i][j][k]
   - (dt/e)*(1.0/dz)*(hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k-1]);
   ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = exy[23][i][j][k] + exz[23][i][j][k];
  }
 }
}

/* calculate PML Ex24 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   exy[24][i][j][k] = exy[24][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[pml_xdim+i][pml_ydim+j][k]-hz[pml_xdim+i][pml_ydim+j-1][k]);
   exz[24][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*exz[24][i][j][k]
   - (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hy[pml_xdim+i][pml_ydim+j][k]-hy[pml_xdim+i][pml_ydim+j][k-1])
   ex[pml_xdim+i][pml_ydim+j][k] = exy[24][i][j][k] + exz[24][i][j][k];
  }
 }
}

/* calculate PML Ex25 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   exy[25][i][j][k] = exy[25][i][j][k]
   + (dt/e)*(1.0/dy)*(hz[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+i][pml_ydim+j-1][pml_zdim+zdim+k]);
   exz[25][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*exz[25][i][j][k]
   - (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hy[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ex[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = exy[25][i][j][k] + exz[25][i][j][k];
  }
 }
}

/* ■■■■ Ey ■■■■ */

/* calculate PML Ey0 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[0][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[0][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][j][k]-hz[i-1][j][k]);
   eyz[0][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[0][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[i][j][k]-hx[i][j][k-1]);
   ey[i][j][k] = eyx[0][i][j][k] + eyz[0][i][j][k];
  }
 }
}

/* calculate PML Ey1 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
```

```
  for(k=1;k<=pml_zdim;k++){
   eyx[1][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[1][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hz[pml_xdim+xdim+i][j][k]-hz[pml_xdim+xdim+i-1][j][k]);
   eyz[1][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[1][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[pml_xdim+xdim+i][j][k]-hx[pml_xdim+xdim+i][j][k-1]);
   ey[pml_xdim+xdim+i][j][k] = eyx[1][i][j][k] + eyz[1][i][j][k];
  }
 }
}

/* calculate PML Ey2 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[2][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[2][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][pml_ydim+ydim+j][k]-hz[i-1][pml_ydim+ydim+j][k]);
   eyz[2][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[2][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[i][pml_ydim+ydim+j][k]-hx[i][pml_ydim+ydim+j][k-1]);
   ey[i][pml_ydim+ydim+j][k] = eyx[2][i][j][k] + eyz[2][i][j][k];
  }
 }
}

/* calculate PML Ey3 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[3][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[3][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hz[pml_xdim+xdim+i-1][pml_ydim+ydim+j][k]);
   eyz[3][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[3][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j][k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = eyx[3][i][j][k] + eyz[3][i][j][k];
  }
 }
}

/* calculate PML Ey4 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[4][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[4][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][j][pml_zdim+zdim+k]-hz[i-1][j][pml_zdim+zdim+k]);
   eyz[4][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[4][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hx[i][j][pml_zdim+zdim+k]-hx[i][j][pml_zdim+zdim+k-1]);
   ey[i][j][pml_zdim+zdim+k] = eyx[4][i][j][k] + eyz[4][i][j][k];
  }
 }
}

/* calculate PML Ey5 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[5][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[5][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i-1][j][pml_zdim+zdim+k]);
   eyz[5][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[5][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][j][pml_zdim+zdim+k-1]);
   ey[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = eyx[5][i][j][k] + eyz[5][i][j][k];
  }
 }
}

/* calculate PML Ey6 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
```

```
   eyx[6][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[6][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   eyz[6][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[6][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ey[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = eyx[6][i][j][k] + eyz[6][i][j][k];
  }
 }
}

/* calculate PML Ey7 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[7][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[7][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   eyz[7][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[7][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = eyx[7][i][j][k] + eyz[7][i][j][k];
  }
 }
}

/* calculate PML Ey8 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[8][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[8][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][j][pml_zdim+k]-hz[i-1][j][pml_zdim+k]);
   eyz[8][i][j][k] = eyz[8][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[i][j][pml_zdim+k]-hx[i][j][pml_zdim+k-1]);
   ey[i][j][pml_zdim+k] = eyx[8][i][j][k] + eyz[8][i][j][k];
  }
 }
}

/* calculate PML Ey9 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[9][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[9][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hz[pml_xdim+xdim+i][j][pml_zdim+k]-hz[pml_xdim+xdim+i-1][j][pml_zdim+k]
   eyz[9][i][j][k] = eyz[9][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[pml_xdim+xdim+i][j][pml_zdim+k]-hx[pml_xdim+xdim+i][j][pml_zdim+k-1]);
   ey[pml_xdim+xdim+i][j][pml_zdim+k] = eyx[9][i][j][k] + eyz[9][i][j][k];
  }
 }
}

/* calculate PML Ey10 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[10][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[10][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[i][pml_ydim+ydim+j][pml_zdim+k]-hz[i-1][pml_ydim+ydim+j][pml_zdim+k]);
   eyz[10][i][j][k] = eyz[10][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[i][pml_ydim+ydim+j][pml_zdim+k]-hx[i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ey[i][pml_ydim+ydim+j][pml_zdim+k] = eyx[10][i][j][k] + eyz[10][i][j][k];
  }
 }
}

/* calculate PML Ey11 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
```

```
   eyx[11][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[11][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hz[pml_xdim+xdim+i-1][pml_ydim+ydim+j][pml_zdim+k]);
   eyz[11][i][j][k] = eyz[11][i][j][k] + (dt/e)*(1.0/dz)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = eyx[11][i][j][k] + eyz[11][i][j][k];
  }
 }
}

/* calculate PML Ey12 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[12][i][j][k] = eyx[12][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][j][k]-hz[pml_xdim+i-1][j][k]);
   eyz[12][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[12][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[pml_xdim+i][j][k]-hx[pml_xdim+i][j][k-1]);
   ey[pml_xdim+i][j][k] = eyx[12][i][j][k] + eyz[12][i][j][k];
  }
 }
}

/* calculate PML Ey13 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[13][i][j][k] = eyx[13][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][j][pml_zdim+zdim+k]-hz[pml_xdim+i-1][j][pml_zdim+zdim+k]);
   eyz[13][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[13][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(hx[pml_xdim+i][j][pml_zdim+zdim+k]-hx[pml_xdim+i][j][pml_zdim+zdim+k-1])
   ey[pml_xdim+i][j][pml_zdim+zdim+k] = eyx[13][i][j][k] + eyz[13][i][j][k];
  }
 }
}

/* calculate PML Ey14 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[14][i][j][k] = eyx[14][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][pml_ydim+ydim+j][k]-hz[pml_xdim+i-1][pml_ydim+ydim+j][k]);
   eyz[14][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[14][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+i][pml_ydim+ydim+j][k]-hx[pml_xdim+i][pml_ydim+ydim+j][k-1]);
   ey[pml_xdim+i][pml_ydim+ydim+j][k] = eyx[14][i][j][k] + eyz[14][i][j][k];
  }
 }
}

/* calculate PML Ey15 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[15][i][j][k] = eyx[15][i][j][k] - (dt/e)*(1.0/dx)
   *(hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   eyz[15][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[15][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k-1]);
   ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = eyx[15][i][j][k] + eyz[15][i][j][k];
  }
 }
}

/* calculate PML Ey16 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[16][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[16][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][pml_ydim+j][k]-hz[i-1][pml_ydim+j][k]);
```

```
   eyz[16][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[16][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[i][pml_ydim+j][k]-hx[i][pml_ydim+j][k-1]);
   ey[i][pml_ydim+j][k] = eyx[16][i][j][k] + eyz[16][i][j][k];
  }
 }
}

/* calculate PML Ey17 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[17][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[17][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+j][k]-hz[pml_xdim+xdim+i-1][pml_ydim+j][k]);
   eyz[17][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[17][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+xdim+i][pml_ydim+j][k]-hx[pml_xdim+xdim+i][pml_ydim+j][k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+j][k] = eyx[17][i][j][k] + eyz[17][i][j][k];
  }
 }
}

/* calculate PML Ey18 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[18][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[18][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[i][pml_ydim+j][pml_zdim+zdim+k]-hz[i-1][pml_ydim+j][pml_zdim+zdim+k]);
   eyz[18][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[18][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[i][pml_ydim+j][pml_zdim+zdim+k]-hx[i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ey[i][pml_ydim+j][pml_zdim+zdim+k] = eyx[18][i][j][k] + eyz[18][i][j][k];
  }
 }
}

/* calculate PML Ey19 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[19][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[19][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+xdim+i-1][pml_ydim+j][pml_zdim+zdim+k]);
   eyz[19][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[19][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = eyx[19][i][j][k] + eyz[19][i][j][k];
  }
 }
}

/* calculate PML Ey20 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[20][i][j][k] = eyx[20][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][j][pml_zdim+k]-hz[pml_xdim+i-1][j][pml_zdim+k]);
   eyz[20][i][j][k] = eyz[20][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[pml_xdim+i][j][pml_zdim+k]-hx[pml_xdim+i][j][pml_zdim+k-1]);
   ey[pml_xdim+i][j][pml_zdim+k] = eyx[20][i][j][k] + eyz[20][i][j][k];
  }
 }
}

/* calculate PML Ey21 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[21][i][j][k] = eyx[21][i][j][k]
```

```
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hz[pml_xdim+i-1][pml_ydim+ydim+j][pml_zdim+k]);
   eyz[21][i][j][k] = eyz[21][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k-1]);
   ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = eyx[21][i][j][k] + eyz[21][i][j][k];
  }
 }
}

/* calculate PML Ey22 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[22][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*eyx[22][i][j][k]
   - (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hz[i][pml_ydim+j][pml_zdim+k]-hz[i-1][pml_ydim+j][pml_zdim+k]);
   eyz[22][i][j][k] = eyz[22][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[i][pml_ydim+j][pml_zdim+k]-hx[i][pml_ydim+j][pml_zdim+k-1]);
   ey[i][pml_ydim+j][pml_zdim+k] = eyx[22][i][j][k] + eyz[22][i][j][k];
  }
 }
}

/* calculate PML Ey23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   eyx[23][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*eyx[23][i][j][k]
   - (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hz[pml_xdim+xdim+i-1][pml_ydim+j][pml_zdim+k]);
   eyz[23][i][j][k] = eyz[23][i][j][k]
   + (dt/e)*(1.0/dz)*(hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k-1]);
   ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = eyx[23][i][j][k] + eyz[23][i][j][k];
  }
 }
}

/* calculate PML Ey24 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   eyx[24][i][j][k] = eyx[24][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][pml_ydim+j][k]-hz[pml_xdim+i-1][pml_ydim+j][k]);
   eyz[24][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*eyz[24][i][j][k]
   + (dt/e)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(hx[pml_xdim+i][pml_ydim+j][k]-hx[pml_xdim+i][pml_ydim+j][k-1]);
   ey[pml_xdim+i][pml_ydim+j][k] = eyx[24][i][j][k] + eyz[24][i][j][k];
  }
 }
}

/* calculate PML Ey25 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   eyx[25][i][j][k] = eyx[25][i][j][k]
   - (dt/e)*(1.0/dx)*(hz[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hz[pml_xdim+i-1][pml_ydim+j][pml_zdim+zdim+k]);
   eyz[25][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*eyz[25][i][j][k]
   + (dt/e)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(hx[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k-1]);
   ey[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = eyx[25][i][j][k] + eyz[25][i][j][k];
  }
 }
}

/* ■■■■ Ez ■■■■ */

/* calculate PML Ez0 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[0][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[0][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][j][k]-hy[i-1][j][k]);
```

```
   ezy[0][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[0][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[i][j][k]-hx[i][j-1][k]);
   ez[i][j][k] = ezx[0][i][j][k] + ezy[0][i][j][k];
  }
 }
}

/* calculate PML Ez1 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[1][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[1][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hy[pml_xdim+xdim+i][j][k]-hy[pml_xdim+xdim+i-1][j][k]);
   ezy[1][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[1][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[pml_xdim+xdim+i][j][k]-hx[pml_xdim+xdim+i][j-1][k]);
   ez[pml_xdim+xdim+i][j][k] = ezx[1][i][j][k] + ezy[1][i][j][k];
  }
 }
}

/* calculate PML Ez2 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[2][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[2][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][pml_ydim+ydim+j][k]-hy[i-1][pml_ydim+ydim+j][k]);
   ezy[2][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[2][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(hx[i][pml_ydim+ydim+j][k]-hx[i][pml_ydim+ydim+j-1][k]);
   ez[i][pml_ydim+ydim+j][k] = ezx[2][i][j][k] + ezy[2][i][j][k];
  }
 }
}

/* calculate PML Ez3 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[3][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[3][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hy[pml_xdim+xdim+i-1][pml_ydim+ydim+j][k]);
   ezy[3][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[3][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j-1][k]);
   ez[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = ezx[3][i][j][k] + ezy[3][i][j][k];
  }
 }
}

/* calculate PML Ez4 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[4][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[4][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][j][pml_zdim+zdim+k]-hy[i-1][j][pml_zdim+zdim+k]);
   ezy[4][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[4][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[i][j][pml_zdim+zdim+k]-hx[i][j-1][pml_zdim+zdim+k]);
   ez[i][j][pml_zdim+zdim+k] = ezx[4][i][j][k] + ezy[4][i][j][k];
  }
 }
}

/* calculate PML Ez5 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[5][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[5][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i-1][j][pml_zdim+zdim+k]);
   ezy[5][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[5][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
```

```
   *(hx[pml_xdim+xdim+i][j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][j-1][pml_zdim+zdim+k]);
   ez[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = ezx[5][i][j][k] + ezy[5][i][j][k];
  }
 }
}

/* calculate PML Ez6 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[6][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[6][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   ezy[6][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[6][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
   ez[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = ezx[6][i][j][k] + ezy[6][i][j][k];
  }
 }
}

/* calculate PML Ez7 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[7][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[7][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   ezy[7][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[7][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
   ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = ezx[7][i][j][k] + ezy[7][i][j][k];
  }
 }
}

/* calculate PML Ez8 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[8][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[8][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][j][pml_zdim+k]-hy[i-1][j][pml_zdim+k]);
   ezy[8][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[8][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[i][j][pml_zdim+k]-hx[i][j-1][pml_zdim+k]);
   ez[i][j][pml_zdim+k] = ezx[8][i][j][k] + ezy[8][i][j][k];
  }
 }
}

/* calculate PML Ez9 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[9][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[9][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][j][pml_zdim+k]-hy[pml_xdim+xdim+i-1][j][pml_zdim+k]);
   ezy[9][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[9][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+xdim+i][j][pml_zdim+k]-hx[pml_xdim+xdim+i][j-1][pml_zdim+k]);
   ez[pml_xdim+xdim+i][j][pml_zdim+k] = ezx[9][i][j][k] + ezy[9][i][j][k];
  }
 }
}

/* calculate PML Ez10 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[10][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[10][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
```

```
   *(hy[i][pml_ydim+ydim+j][pml_zdim+k]-hy[i-1][pml_ydim+ydim+j][pml_zdim+k]);
   ezy[10][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[10][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[i][pml_ydim+ydim+j][pml_zdim+k]-hx[i][pml_ydim+ydim+j-1][pml_zdim+k]);
   ez[i][pml_ydim+ydim+j][pml_zdim+k] = ezx[10][i][j][k] + ezy[10][i][j][k];
  }
 }
}

/* calculate PML Ez11 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[11][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[11][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hy[pml_xdim+xdim+i-1][pml_ydim+ydim+j][pml_zdim+k]);
   ezy[11][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[11][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+ydim+j-1][pml_zdim+k]);
   ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = ezx[11][i][j][k] + ezy[11][i][j][k];
  }
 }
}

/* calculate PML Ez12 *//*●*/
for(i=1;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[12][i][j][k] = ezx[12][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][j][k]-hy[pml_xdim+i-1][j][k]);
   ezy[12][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[12][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[pml_xdim+i][j][k]-hx[pml_xdim+i][j-1][k]);
   ez[pml_xdim+i][j][k] = ezx[12][i][j][k] + ezy[12][i][j][k];
  }
 }
}

/* calculate PML Ez13 *//*●*/
for(i=1;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[13][i][j][k] = ezx[13][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][j][pml_zdim+zdim+k]-hy[pml_xdim+i-1][j][pml_zdim+zdim+k]);
   ezy[13][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[13][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+i][j][pml_zdim+zdim+k]-hx[pml_xdim+i][j-1][pml_zdim+zdim+k]);
   ez[pml_xdim+i][j][pml_zdim+zdim+k] = ezx[13][i][j][k] + ezy[13][i][j][k];
  }
 }
}

/* calculate PML Ez14 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[14][i][j][k] = ezx[14][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][pml_ydim+ydim+j][k]-hy[pml_xdim+i-1][pml_ydim+ydim+j][k]);
   ezy[14][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[14][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(hx[pml_xdim+i][pml_ydim+ydim+j][k]-hx[pml_xdim+i][pml_ydim+ydim+j-1][k]
   ez[pml_xdim+i][pml_ydim+ydim+j][k] = ezx[14][i][j][k] + ezy[14][i][j][k];
  }
 }
}

/* calculate PML Ez15 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[15][i][j][k] = ezx[15][i][j][k] + (dt/e)*(1.0/dx)
   *(hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+i-1][pml_ydim+ydim+j][pml_zdim+zdim+k]);
```

```
    ezy[15][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[15][i][j][k]
    - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+i][pml_ydim+ydim+j-1][pml_zdim+zdim+k]);
    ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = ezx[15][i][j][k] + ezy[15][i][j][k];
   }
  }
}

/* calculate PML Ez16 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[16][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[16][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][pml_ydim+j][k]-hy[i-1][pml_ydim+j][k]);
   ezy[16][i][j][k] = ezy[16][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[i][pml_ydim+j][k]-hx[i][pml_ydim+j-1][k]);
   ez[i][pml_ydim+j][k] = ezx[16][i][j][k] + ezy[16][i][j][k];
  }
 }
}

/* calculate PML Ez17 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[17][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[17][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(hy[pml_xdim+xdim+i][pml_ydim+j][k]-hy[pml_xdim+xdim+i-1][pml_ydim+j][k]
   ezy[17][i][j][k] = ezy[17][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[pml_xdim+xdim+i][pml_ydim+j][k]-hx[pml_xdim+xdim+i][pml_ydim+j-1][k]);
   ez[pml_xdim+xdim+i][pml_ydim+j][k] = ezx[17][i][j][k] + ezy[17][i][j][k];
  }
 }
}

/* calculate PML Ez18 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[18][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[18][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[i][pml_ydim+j][pml_zdim+zdim+k]-hy[i-1][pml_ydim+j][pml_zdim+zdim+k]);
   ezy[18][i][j][k] = ezy[18][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[i][pml_ydim+j][pml_zdim+zdim+k]-hx[i][pml_ydim+j-1][pml_zdim+zdim+k]);
   ez[i][pml_ydim+j][pml_zdim+zdim+k] = ezx[18][i][j][k] + ezy[18][i][j][k];
  }
 }
}

/* calculate PML Ez19 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[19][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[19][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+xdim+i-1][pml_ydim+j][pml_zdim+zdim+k]);
   ezy[19][i][j][k] = ezy[19][i][j][k] - (dt/e)*(1.0/dy)
   *(hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+j-1][pml_zdim+zdim+k]);
   ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = ezx[19][i][j][k] + ezy[19][i][j][k];
  }
 }
}

/* calculate PML Ez20 *//*●*/
for(i=1;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[20][i][j][k] = ezx[20][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][j][pml_zdim+k]-hy[pml_xdim+i-1][j][pml_zdim+k]);
   ezy[20][i][j][k] = (1.0-(roy[pml_ydim-j]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*ezy[20][i][j][k]
   - (dt/e)/(1.0+(roy[pml_ydim-j]*dt)/(2.0*e))*(1.0/dy)*(hx[pml_xdim+i][j][pml_zdim+k]-hx[pml_xdim+i][j-1][pml_zdim+k])
```

```
   ez[pml_xdim+i][j][pml_zdim+k] = ezx[20][i][j][k] + ezy[20][i][j][k];
  }
 }
}

/* calculate PML Ez21 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[21][i][j][k] = ezx[21][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hy[pml_xdim+i-1][pml_ydim+ydim+j][pml_zdim+k]);
   ezy[21][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*ezy[21][i][j][k]
   - (dt/e)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]-hx[pml_xdim+i][pml_ydim+ydim+j-1][pml_zdim+k]);
   ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = ezx[21][i][j][k] + ezy[21][i][j][k];
  }
 }
}

/* calculate PML Ez22 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[22][i][j][k] = (1.0-(rox[pml_xdim-i]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*ezx[22][i][j][k]
   + (dt/e)/(1.0+(rox[pml_xdim-i]*dt)/(2.0*e))*(1.0/dx)*(hy[i][pml_ydim+j][pml_zdim+k]-hy[i-1][pml_ydim+j][pml_zdim+k]);
   ezy[22][i][j][k] = ezy[22][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[i][pml_ydim+j][pml_zdim+k]-hx[i][pml_ydim+j-1][pml_zdim+k]);
   ez[i][pml_ydim+j][pml_zdim+k] = ezx[22][i][j][k] + ezy[22][i][j][k];
  }
 }
}

/* calculate PML Ez23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   ezx[23][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*ezx[23][i][j][k]
   + (dt/e)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hy[pml_xdim+xdim+i-1][pml_ydim+j][pml_zdim+k]);
   ezy[23][i][j][k] = ezy[23][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]-hx[pml_xdim+xdim+i][pml_ydim+j-1][pml_zdim+k]);
   ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = ezx[23][i][j][k] + ezy[23][i][j][k];
  }
 }
}

/* calculate PML Ez24 *//*●*/
for(i=1;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   ezx[24][i][j][k] = ezx[24][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][pml_ydim+j][k]-hy[pml_xdim+i-1][pml_ydim+j][k]);
   ezy[24][i][j][k] = ezy[24][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[pml_xdim+i][pml_ydim+j][k]-hx[pml_xdim+i][pml_ydim+j-1][k]);
   ez[pml_xdim+i][pml_ydim+j][k] = ezx[24][i][j][k] + ezy[24][i][j][k];
  }
 }
}

/* calculate PML Ez25 *//*●*/
for(i=1;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   ezx[25][i][j][k] = ezx[25][i][j][k]
   + (dt/e)*(1.0/dx)*(hy[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hy[pml_xdim+i-1][pml_ydim+j][pml_zdim+zdim+k]);
   ezy[25][i][j][k] = ezy[25][i][j][k]
   - (dt/e)*(1.0/dy)*(hx[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]-hx[pml_xdim+i][pml_ydim+j-1][pml_zdim+zdim+k]);
   ez[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = ezx[25][i][j][k] + ezy[25][i][j][k];
  }
 }
```

```
}

/* ■■■■■■■■■■■■■■■■■■ PML for E end ■■■■■■■■■■■■■■■■■■ */

/* ■■■■■■■■■■■■■■■■■■ soft source for E ■■■■■■■■■■■■■■■■■■ */

 ez[pml_xdim+f_x[i]][pml_ydim+f_y[i]][pml_zdim+f_z[i]] = sin(2.0*3.14*f[i]*t);
}

/* ■■■■■■■■■■■■■■■■■■ renew time step ■■■■■■■■■■■■■■■■■■ */

/* 半ステップ更新 *//*●*/
t += (dt/2.0);

/* calculate Hx *//*●*/
for(i=pml_xdim+1;i<pml_xdim+xdim;i++){
 for(j=pml_ydim;j<pml_ydim+ydim;j++){
  for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
   hx[i][j][k] = hx[i][j][k]
   - (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dy)*(ez[i][j+1][k]-ez[i][j][k])
   + (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dz)*(ey[i][j][k+1]-ey[i][j][k]);
  }
 }
}

/* calculate Hy *//*●*/
for(i=pml_xdim;i<pml_xdim+xdim;i++){
 for(j=pml_ydim+1;j<pml_ydim+ydim;j++){
  for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
   hy[i][j][k] = hy[i][j][k]
   - (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dz)*(ex[i][j][k+1]-ex[i][j][k])
   + (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dx)*(ez[i+1][j][k]-ez[i][j][k]);
  }
 }
}

/* calculate Hz *//*●*/
for(i=pml_xdim;i<pml_xdim+xdim;i++){
 for(j=pml_ydim;j<pml_ydim+ydim;j++){
  for(k=pml_zdim+1;k<pml_zdim+zdim;k++){
   hz[i][j][k] = hz[i][j][k]
   - (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dx)*(ey[i+1][j][k]-ey[i][j][k])
   + (dt/ID_data_struct.mu[(int)ID[i-(pml_xdim)][j-(pml_ydim)][k-(pml_zdim)]])*(1/dy)*(ex[i][j+1][k]-ex[i][j][k]);
  }
 }
}

/* ■■■■■■■■■■■■■■■■■■ PML for H ■■■■■■■■■■■■■■■■■■ */

/* ■■■■ Hx ■■■■ */

/* calculate PML Hx0 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[0][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[0][i][j][k]
   - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[i][j+1][k]-ez[i][j][k]);
   hxz[0][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[0][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][j][k+1]-ey[i][j][k]);
   hx[i][j][k] = hxy[0][i][j][k] + hxz[0][i][j][k];
  }
 }
}

/* calculate PML Hx1 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[1][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[1][i][j][k]
   - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[pml_xdim+xdim+i][j+1][k]-ez[pml_xdim+xdim+i][j][k]);
```

```
   hxz[1][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[1][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[pml_xdim+xdim+i][j][k+1]-ey[pml_xdim+xdim+i][j][k]);
   hx[pml_xdim+xdim+i][j][k] = hxy[1][i][j][k] + hxz[1][i][j][k];
  }
 }
}

/* calculate PML Hx2 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[2][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[2][i][j][k]
   - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ez[i][pml_ydim+ydim+j+1][k]-ez[i][pml_ydim+ydim+j][k]);
   hxz[2][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[2][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][pml_ydim+ydim+j][k+1]-ey[i][pml_ydim+ydim+j][k]);
   hx[i][pml_ydim+ydim+j][k] = hxy[2][i][j][k] + hxz[2][i][j][k];
  }
 }
}

/* calculate PML Hx3 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[3][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[3][i][j][k]
   - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ez[pml_xdim+xdim+i][pml_ydim+ydim+j+1][k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
   hxz[3][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[3][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(ey[pml_xdim+xdim+i][pml_ydim+ydim+j][k+1]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
   hx[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = hxy[3][i][j][k] + hxz[3][i][j][k];
  }
 }
}

/* calculate PML Hx4 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[4][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[4][i][j][k]
   - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[i][j+1][pml_zdim+zdim+k]-ez[i][j][pml_zdim+zdim+k]);
   hxz[4][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[4][i][j][k]
   + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][j][pml_zdim+zdim+k+1]-ey[i][j][pml_zdim+zdim+k]);
   hx[i][j][pml_zdim+zdim+k] = hxy[4][i][j][k] + hxz[4][i][j][k];
  }
 }
}

/* calculate PML Hx5 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[5][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[5][i][j][k]
   - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
   *(ez[pml_xdim+xdim+i][j+1][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hxz[5][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[5][i][j][k]
   + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ey[pml_xdim+xdim+i][j][pml_zdim+zdim+k+1]-ey[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hx[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = hxy[5][i][j][k] + hxz[5][i][j][k];
  }
 }
}

/* calculate PML Hx6 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[6][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[6][i][j][k]
   - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ez[i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ez[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
```

```
    hxz[6][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[6][i][j][k]
    + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
    *(ey[i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ey[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hx[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hxy[6][i][j][k] + hxz[6][i][j][k];
   }
  }
 }

 /* calculate PML Hx7 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hxy[7][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[7][i][j][k]
    - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ez[pml_xdim+xdim+i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hxz[7][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[7][i][j][k]
    + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
    *(ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hxy[7][i][j][k] + hxz[7][i][j][k];
   }
  }
 }

 /* calculate PML Hx8 *//*●*/
 for(i=1;i<=pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hxy[8][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[8][i][j][k]
    - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[i][j+1][pml_zdim+k]-ez[i][j][pml_zdim+k]);
    hxz[8][i][j][k] = hxz[8][i][j][k]
    + (dt/m)*(1.0/dz)*(ey[i][j][pml_zdim+k+1]-ey[i][j][pml_zdim+k]);
    hx[i][j][pml_zdim+k] = hxy[8][i][j][k] + hxz[8][i][j][k];
   }
  }
 }

 /* calculate PML Hx9 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hxy[9][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[9][i][j][k]
    - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
    *(ez[pml_xdim+xdim+i][j+1][pml_zdim+k]-ez[pml_xdim+xdim+i][j][pml_zdim+k]);
    hxz[9][i][j][k] = hxz[9][i][j][k]
    + (dt/m)*(1.0/dz)*(ey[pml_xdim+xdim+i][j][pml_zdim+k+1]-ey[pml_xdim+xdim+i][j][pml_zdim+k]);
    hx[pml_xdim+xdim+i][j][pml_zdim+k] = hxy[9][i][j][k] + hxz[9][i][j][k];
   }
  }
 }

 /* calculate PML Hx10 *//*●*/
 for(i=1;i<=pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hxy[10][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[10][i][j][k]
    - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ez[i][pml_ydim+ydim+j+1][pml_zdim+k]-ez[i][pml_ydim+ydim+j][pml_zdim+k])
    hxz[10][i][j][k] = hxz[10][i][j][k]
    + (dt/m)*(1.0/dz)*(ey[i][pml_ydim+ydim+j][pml_zdim+k+1]-ey[i][pml_ydim+ydim+j][pml_zdim+k]);
    hx[i][pml_ydim+ydim+j][pml_zdim+k] = hxy[10][i][j][k] + hxz[10][i][j][k];
   }
  }
 }

 /* calculate PML Hx11 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hxy[11][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[11][i][j][k]
    - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ez[pml_xdim+xdim+i][pml_ydim+ydim+j+1][pml_zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
```

```
    hxz[11][i][j][k] = hxz[11][i][j][k] + (dt/m)*(1.0/dz)
    *(ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k+1]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
    hx[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hxy[11][i][j][k] + hxz[11][i][j][k];
   }
  }
 }

 /* calculate PML Hx12 *//*●*/
 for(i=1;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hxy[12][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[12][i][j][k]
    - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[pml_xdim+i][j+1][k]-ez[pml_xdim+i][j][k]);
    hxz[12][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[12][i][j][k]
    + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[pml_xdim+i][j][k+1]-ey[pml_xdim+i][j][k]);
    hx[pml_xdim+i][j][k] = hxy[12][i][j][k] + hxz[12][i][j][k];
   }
  }
 }

 /* calculate PML Hx13 *//*●*/
 for(i=1;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hxy[13][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[13][i][j][k]
    - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
    *(ez[pml_xdim+i][j+1][pml_zdim+zdim+k]-ez[pml_xdim+i][j][pml_zdim+zdim+k]);
    hxz[13][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[13][i][j][k]
    + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ey[pml_xdim+i][j][pml_zdim+zdim+k+1]-ey[pml_xdim+i][j][pml_zdim+zdim+k]);
    hx[pml_xdim+i][j][pml_zdim+zdim+k] = hxy[13][i][j][k] + hxz[13][i][j][k];
   }
  }
 }

 /* calculate PML Hx14 *//*●*/
 for(i=1;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hxy[14][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[14][i][j][k]
    - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ez[pml_xdim+i][pml_ydim+ydim+j+1][k]-ez[pml_xdim+i][pml_ydim+ydim+j][k]);
    hxz[14][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[14][i][j][k]
    + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
    *(ey[pml_xdim+i][pml_ydim+ydim+j][k+1]-ey[pml_xdim+i][pml_ydim+ydim+j][k]);
    hx[pml_xdim+i][pml_ydim+ydim+j][k] = hxy[14][i][j][k] + hxz[14][i][j][k];
   }
  }
 }

 /* calculate PML Hx15 *//*●*/
 for(i=1;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hxy[15][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[15][i][j][k]
    - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ez[pml_xdim+i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hxz[15][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[15][i][j][k]
    + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
    *(ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hxy[15][i][j][k] + hxz[15][i][j][k];
   }
  }
 }

 /* calculate PML Hx16 *//*●*/
 for(i=1;i<=pml_xdim;i++){
  for(j=0;j<ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hxy[16][i][j][k] = hxy[16][i][j][k]
    - (dt/m)*(1.0/dy)*(ez[i][pml_ydim+j+1][k]-ez[i][pml_ydim+j][k]);
    hxz[16][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[16][i][j][k]
```

```
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][pml_ydim+j][k+1]-ey[i][pml_ydim+j][k]);
   hx[i][pml_ydim+j][k] = hxy[16][i][j][k] + hxz[16][i][j][k];
  }
 }
}

/* calculate PML Hx17 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[17][i][j][k] = hxy[17][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[pml_xdim+xdim+i][pml_ydim+j+1][k]-ez[pml_xdim+xdim+i][pml_ydim+j][k]);
   hxz[17][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[17][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(ey[pml_xdim+xdim+i][pml_ydim+j][k+1]-ey[pml_xdim+xdim+i][pml_ydim+j][k]);
   hx[pml_xdim+xdim+i][pml_ydim+j][k] = hxy[17][i][j][k] + hxz[17][i][j][k];
  }
 }
}

/* calculate PML Hx18 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[18][i][j][k] = hxy[18][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[i][pml_ydim+j+1][pml_zdim+zdim+k]-ez[i][pml_ydim+j][pml_zdim+zdim+k]);
   hxz[18][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[18][i][j][k]
   + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ey[i][pml_ydim+j][pml_zdim+zdim+k+1]-ey[i][pml_ydim+j][pml_zdim+zdim+k]);
   hx[i][pml_ydim+j][pml_zdim+zdim+k] = hxy[18][i][j][k] + hxz[18][i][j][k];
  }
 }
}

/* calculate PML Hx19 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[19][i][j][k] = hxy[19][i][j][k] - (dt/m)*(1.0/dy)
   *(ez[pml_xdim+xdim+i][pml_ydim+j+1][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hxz[19][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[19][i][j][k]
   + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k+1]-ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hxy[19][i][j][k] + hxz[19][i][j][k];
  }
 }
}

/* calculate PML Hx20 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hxy[20][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hxy[20][i][j][k]
   - (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ez[pml_xdim+i][j+1][pml_zdim+k]-ez[pml_xdim+i][j][pml_zdim+k])
   hxz[20][i][j][k] = hxz[20][i][j][k]
   + (dt/m)*(1.0/dz)*(ey[pml_xdim+i][j][pml_zdim+k+1]-ey[pml_xdim+i][j][pml_zdim+k]);
   hx[pml_xdim+i][j][pml_zdim+k] = hxy[20][i][j][k] + hxz[20][i][j][k];
  }
 }
}

/* calculate PML Hx21 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hxy[21][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hxy[21][i][j][k]
   - (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ez[pml_xdim+i][pml_ydim+ydim+j+1][pml_zdim+k]-ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hxz[21][i][j][k] = hxz[21][i][j][k]
   + (dt/m)*(1.0/dz)*(ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k+1]-ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hx[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hxy[21][i][j][k] + hxz[21][i][j][k];
```

```
  }
 }
}

/* calculate PML Hx22 *//*●*/
for(i=1;i<=pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hxy[22][i][j][k] = hxy[22][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[i][pml_ydim+j+1][pml_zdim+k]-ez[i][pml_ydim+j][pml_zdim+k]);
   hxz[22][i][j][k] = hxz[22][i][j][k]
   + (dt/m)*(1.0/dz)*(ey[i][pml_ydim+j][pml_zdim+k+1]-ey[i][pml_ydim+j][pml_zdim+k]);
   hx[i][pml_ydim+j][pml_zdim+k] = hxy[22][i][j][k] + hxz[22][i][j][k];
  }
 }
}

/* calculate PML Hx23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hxy[23][i][j][k] = hxy[23][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[pml_xdim+xdim+i][pml_ydim+j+1][pml_zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hxz[23][i][j][k] = hxz[23][i][j][k]
   + (dt/m)*(1.0/dz)*(ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k+1]-ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hx[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = hxy[23][i][j][k] + hxz[23][i][j][k];
  }
 }
}

/* calculate PML Hx24 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hxy[24][i][j][k] = hxy[24][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[pml_xdim+i][pml_ydim+j+1][k]-ez[pml_xdim+i][pml_ydim+j][k]);
   hxz[24][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hxz[24][i][j][k]
   + (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ey[pml_xdim+i][pml_ydim+j][k+1]-ey[pml_xdim+i][pml_ydim+j][k]);
   hx[pml_xdim+i][pml_ydim+j][k] = hxy[24][i][j][k] + hxz[24][i][j][k];
  }
 }
}

/* calculate PML Hx25 *//*●*/
for(i=1;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hxy[25][i][j][k] = hxy[25][i][j][k]
   - (dt/m)*(1.0/dy)*(ez[pml_xdim+i][pml_ydim+j+1][pml_zdim+zdim+k]-ez[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hxz[25][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hxz[25][i][j][k]
   + (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ey[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k+1]-ey[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hx[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hxy[25][i][j][k] + hxz[25][i][j][k];
  }
 }
}

/* ■■■■ Hy ■■■■ */

/* calculate PML Hy0 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[0][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[0][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][j][k]-ez[i][j][k]);
   hyz[0][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[0][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][j][k+1]-ex[i][j][k]);
   hy[i][j][k] = hyx[0][i][j][k] + hyz[0][i][j][k];
  }
 }
```

```
}

/* calculate PML Hy1 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[1][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[1][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ez[pml_xdim+xdim+i+1][j][k]-ez[pml_xdim+xdim+i][j][k]);
   hyz[1][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[1][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[pml_xdim+xdim+i][j][k+1]-ex[pml_xdim+xdim+i][j][k]);
   hy[pml_xdim+xdim+i][j][k] = hyx[1][i][j][k] + hyz[1][i][j][k];
  }
 }
}

/* calculate PML Hy2 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[2][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[2][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][pml_ydim+ydim+j][k]-ez[i][pml_ydim+ydim+j][k]);
   hyz[2][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[2][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][pml_ydim+ydim+j][k+1]-ex[i][pml_ydim+ydim+j][k]);
   hy[i][pml_ydim+ydim+j][k] = hyx[2][i][j][k] + hyz[2][i][j][k];
  }
 }
}

/* calculate PML Hy3 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[3][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[3][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][pml_ydim+ydim+j][k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
   hyz[3][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[3][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j][k+1]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
   hy[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = hyx[3][i][j][k] + hyz[3][i][j][k];
  }
 }
}

/* calculate PML Hy4 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[4][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[4][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][j][pml_zdim+zdim+k]-ez[i][j][pml_zdim+zdim+k]);
   hyz[4][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[4][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][j][pml_zdim+zdim+k+1]-ex[i][j][pml_zdim+zdim+k]);
   hy[i][j][pml_zdim+zdim+k] = hyx[4][i][j][k] + hyz[4][i][j][k];
  }
 }
}

/* calculate PML Hy5 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[5][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[5][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][j][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hyz[5][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[5][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+xdim+i][j][pml_zdim+zdim+k+1]-ex[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hy[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = hyx[5][i][j][k] + hyz[5][i][j][k];
  }
 }
}
```

```
/* calculate PML Hy6 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[6][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[6][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
   *(ez[i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ez[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hyz[6][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[6][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ex[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hy[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hyx[6][i][j][k] + hyz[6][i][j][k];
  }
 }
}

/* calculate PML Hy7 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[7][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[7][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hyz[7][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[7][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hyx[7][i][j][k] + hyz[7][i][j][k];
  }
 }
}

/* calculate PML Hy8 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[8][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[8][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][j][pml_zdim+k]-ez[i][j][pml_zdim+k]);
   hyz[8][i][j][k] = hyz[8][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[i][j][pml_zdim+k+1]-ex[i][j][pml_zdim+k]);
   hy[i][j][pml_zdim+k] = hyx[8][i][j][k] + hyz[8][i][j][k];
  }
 }
}

/* calculate PML Hy9 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[9][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[9][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ez[pml_xdim+xdim+i+1][j][pml_zdim+k]-ez[pml_xdim+xdim+i][j][pml_zdim+k]);
   hyz[9][i][j][k] = hyz[9][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[pml_xdim+xdim+i][j][pml_zdim+k+1]-ex[pml_xdim+xdim+i][j][pml_zdim+k]);
   hy[pml_xdim+xdim+i][j][pml_zdim+k] = hyx[9][i][j][k] + hyz[9][i][j][k];
  }
 }
}

/* calculate PML Hy10 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[10][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[10][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
   *(ez[i+1][pml_ydim+ydim+j][pml_zdim+k]-ez[i][pml_ydim+ydim+j][pml_zdim+k]);
   hyz[10][i][j][k] = hyz[10][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[i][pml_ydim+ydim+j][pml_zdim+k+1]-ex[i][pml_ydim+ydim+j][pml_zdim+k]);
   hy[i][pml_ydim+ydim+j][pml_zdim+k] = hyx[10][i][j][k] + hyz[10][i][j][k];
  }
 }
}
```

```
/* calculate PML Hy11 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[11][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[11][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][pml_ydim+ydim+j][pml_zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hyz[11][i][j][k] = hyz[11][i][j][k] - (dt/m)*(1.0/dz)
   *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k+1]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hy[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hyx[11][i][j][k] + hyz[11][i][j][k];
  }
 }
}

/* calculate PML Hy12 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[12][i][j][k] = hyx[12][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][j][k]-ez[pml_xdim+i][j][k]);
   hyz[12][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[12][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[pml_xdim+i][j][k+1]-ex[pml_xdim+i][j][k]);
   hy[pml_xdim+i][j][k] = hyx[12][i][j][k] + hyz[12][i][j][k];
  }
 }
}

/* calculate PML Hy13 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[13][i][j][k] = hyx[13][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][j][pml_zdim+zdim+k]-ez[pml_xdim+i][j][pml_zdim+zdim+k]);
   hyz[13][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[13][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ex[pml_xdim+i][j][pml_zdim+zdim+k+1]-ex[pml_xdim+i][j][pml_zdim+zdim+k]);
   hy[pml_xdim+i][j][pml_zdim+zdim+k] = hyx[13][i][j][k] + hyz[13][i][j][k];
  }
 }
}

/* calculate PML Hy14 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[14][i][j][k] = hyx[14][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][pml_ydim+ydim+j][k]-ez[pml_xdim+i][pml_ydim+ydim+j][k]);
   hyz[14][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[14][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+i][pml_ydim+ydim+j][k+1]-ex[pml_xdim+i][pml_ydim+ydim+j][k]);
   hy[pml_xdim+i][pml_ydim+ydim+j][k] = hyx[14][i][j][k] + hyz[14][i][j][k];
  }
 }
}

/* calculate PML Hy15 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[15][i][j][k] = hyx[15][i][j][k] + (dt/m)*(1.0/dx)
   *(ez[pml_xdim+i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hyz[15][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[15][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k+1]-ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hyx[15][i][j][k] + hyz[15][i][j][k];
  }
 }
}

/* calculate PML Hy16 *//*●*/
```

```
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[16][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[16][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][pml_ydim+j][k]-ez[i][pml_ydim+j][k]);
   hyz[16][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[16][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][pml_ydim+j][k+1]-ex[i][pml_ydim+j][k]);
   hy[i][pml_ydim+j][k] = hyx[16][i][j][k] + hyz[16][i][j][k];
  }
 }
}

/* calculate PML Hy17 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[17][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[17][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ez[pml_xdim+xdim+i+1][pml_ydim+j][k]-ez[pml_xdim+xdim+i][pml_ydim+j][k]);
   hyz[17][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[17][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+xdim+i][pml_ydim+j][k+1]-ex[pml_xdim+xdim+i][pml_ydim+j][k]);
   hy[pml_xdim+xdim+i][pml_ydim+j][k] = hyx[17][i][j][k] + hyz[17][i][j][k];
  }
 }
}

/* calculate PML Hy18 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[18][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[18][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
   *(ez[i+1][pml_ydim+j][pml_zdim+zdim+k]-ez[i][pml_ydim+j][pml_zdim+zdim+k]);
   hyz[18][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[18][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)*(ex[i][pml_ydim+j][pml_zdim+zdim+k+1]-ex[i][pml_ydim+j][pml_zdim+zdim+k]);
   hy[i][pml_ydim+j][pml_zdim+zdim+k] = hyx[18][i][j][k] + hyz[18][i][j][k];
  }
 }
}

/* calculate PML Hy19 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[19][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[19][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][pml_ydim+j][pml_zdim+zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hyz[19][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[19][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k+1]-ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hyx[19][i][j][k] + hyz[19][i][j][k];
  }
 }
}

/* calculate PML Hy20 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<=pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[20][i][j][k] = hyx[20][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][j][pml_zdim+k]-ez[pml_xdim+i][j][pml_zdim+k]);
   hyz[20][i][j][k] = hyz[20][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[pml_xdim+i][j][pml_zdim+k+1]-ex[pml_xdim+i][j][pml_zdim+k]);
   hy[pml_xdim+i][j][pml_zdim+k] = hyx[20][i][j][k] + hyz[20][i][j][k];
  }
 }
}

/* calculate PML Hy21 *//*●*/
for(i=0;i<xdim;i++){
```

```
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[21][i][j][k] = hyx[21][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][pml_ydim+ydim+j][pml_zdim+k]-ez[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hyz[21][i][j][k] = hyz[21][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k+1]-ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hy[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hyx[21][i][j][k] + hyz[21][i][j][k];
  }
 }
}

/* calculate PML Hy22 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[22][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hyx[22][i][j][k]
   + (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ez[i+1][pml_ydim+j][pml_zdim+k]-ez[i][pml_ydim+j][pml_zdim+k]);
   hyz[22][i][j][k] = hyz[22][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[i][pml_ydim+j][pml_zdim+k+1]-ex[i][pml_ydim+j][pml_zdim+k]);
   hy[i][pml_ydim+j][pml_zdim+k] = hyx[22][i][j][k] + hyz[22][i][j][k];
  }
 }
}

/* calculate PML Hy23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hyx[23][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hyx[23][i][j][k]
   + (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ez[pml_xdim+xdim+i+1][pml_ydim+j][pml_zdim+k]-ez[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hyz[23][i][j][k] = hyz[23][i][j][k]
   - (dt/m)*(1.0/dz)*(ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k+1]-ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hy[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = hyx[23][i][j][k] + hyz[23][i][j][k];
  }
 }
}

/* calculate PML Hy24 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hyx[24][i][j][k] = hyx[24][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][pml_ydim+j][k]-ez[pml_xdim+i][pml_ydim+j][k]);
   hyz[24][i][j][k] = (1.0-(roz[pml_zdim-k]*dt)/(2.0*e))/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*hyz[24][i][j][k]
   - (dt/m)/(1.0+(roz[pml_zdim-k]*dt)/(2.0*e))*(1.0/dz)*(ex[pml_xdim+i][pml_ydim+j][k+1]-ex[pml_xdim+i][pml_ydim+j][k]);
   hy[pml_xdim+i][pml_ydim+j][k] = hyx[24][i][j][k] + hyz[24][i][j][k];
  }
 }
}

/* calculate PML Hy25 *//*●*/
for(i=0;i<xdim;i++){
 for(j=1;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hyx[25][i][j][k] = hyx[25][i][j][k]
   + (dt/m)*(1.0/dx)*(ez[pml_xdim+i+1][pml_ydim+j][pml_zdim+zdim+k]-ez[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hyz[25][i][j][k] = (1.0-(roz[k]*dt)/(2.0*e))/(1.0+(roz[k]*dt)/(2.0*e))*hyz[25][i][j][k]
   - (dt/m)/(1.0+(roz[k]*dt)/(2.0*e))*(1.0/dz)
   *(ex[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k+1]-ex[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hy[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hyx[25][i][j][k] + hyz[25][i][j][k];
  }
 }
}

/* ■■■■ Hz ■■■■ *//*●*/

/* calculate PML Hz0 */
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
```

```
   for(k=1;k<=pml_zdim;k++){
    hzx[0][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[0][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][j][k]-ey[i][j][k]);
    hzy[0][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[0][i][j][k]
    + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[i][j+1][k]-ex[i][j][k]);
    hz[i][j][k] = hzx[0][i][j][k] + hzy[0][i][j][k];
   }
  }
 }

 /* calculate PML Hz1 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[1][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[1][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ey[pml_xdim+xdim+i+1][j][k]-ey[pml_xdim+xdim+i][j][k]);
    hzy[1][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[1][i][j][k]
    + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[pml_xdim+xdim+i][j+1][k]-ex[pml_xdim+xdim+i][j][k]);
    hz[pml_xdim+xdim+i][j][k] = hzx[1][i][j][k] + hzy[1][i][j][k];
   }
  }
 }

 /* calculate PML Hz2 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[2][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[2][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][pml_ydim+ydim+j][k]-ey[i][pml_ydim+ydim+j][k]);
    hzy[2][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[2][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ex[i][pml_ydim+ydim+j+1][k]-ex[i][pml_ydim+ydim+j][k]);
    hz[i][pml_ydim+ydim+j][k] = hzx[2][i][j][k] + hzy[2][i][j][k];
   }
  }
 }

 /* calculate PML Hz3 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[3][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[3][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
    *(ey[pml_xdim+xdim+i+1][pml_ydim+ydim+j][k]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
    hzy[3][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[3][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j+1][k]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][k]);
    hz[pml_xdim+xdim+i][pml_ydim+ydim+j][k] = hzx[3][i][j][k] + hzy[3][i][j][k];
   }
  }
 }

 /* calculate PML Hz4 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[4][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[4][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][j][pml_zdim+zdim+k]-ey[i][j][pml_zdim+zdim+k]);
    hzy[4][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[4][i][j][k]
    + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[i][j+1][pml_zdim+zdim+k]-ex[i][j][pml_zdim+zdim+k]);
    hz[i][j][pml_zdim+zdim+k] = hzx[4][i][j][k] + hzy[4][i][j][k];
   }
  }
 }

 /* calculate PML Hz5 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[5][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[5][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
```

```
   *(ey[pml_xdim+xdim+i+1][j][pml_zdim+zdim+k]-ey[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hzy[5][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[5][i][j][k]
   + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
   *(ex[pml_xdim+xdim+i][j+1][pml_zdim+zdim+k]-ex[pml_xdim+xdim+i][j][pml_zdim+zdim+k]);
   hz[pml_xdim+xdim+i][j][pml_zdim+zdim+k] = hzx[5][i][j][k] + hzy[5][i][j][k];
  }
 }
}

/* calculate PML Hz6 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hzx[6][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[6][i][j][k]
   - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
   *(ey[i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ey[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hzy[6][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[6][i][j][k]
   + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ex[i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ex[i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hz[i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hzx[6][i][j][k] + hzy[6][i][j][k];
  }
 }
}

/* calculate PML Hz7 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hzx[7][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[7][i][j][k]
   - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ey[pml_xdim+xdim+i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hzy[7][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[7][i][j][k]
   + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
   hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hzx[7][i][j][k] + hzy[7][i][j][k];
  }
 }
}

/* calculate PML Hz8 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hzx[8][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[8][i][j][k]
   - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][j][pml_zdim+k]-ey[i][j][pml_zdim+k]);
   hzy[8][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[8][i][j][k]
   + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[i][j+1][pml_zdim+k]-ex[i][j][pml_zdim+k]);
   hz[i][j][pml_zdim+k] = hzx[8][i][j][k] + hzy[8][i][j][k];
  }
 }
}

/* calculate PML Hz9 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hzx[9][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[9][i][j][k]
   - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ey[pml_xdim+xdim+i+1][j][pml_zdim+k]-ey[pml_xdim+xdim+i][j][pml_zdim+k]);
   hzy[9][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[9][i][j][k]
   + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
   *(ex[pml_xdim+xdim+i][j+1][pml_zdim+k]-ex[pml_xdim+xdim+i][j][pml_zdim+k]);
   hz[pml_xdim+xdim+i][j][pml_zdim+k] = hzx[9][i][j][k] + hzy[9][i][j][k];
  }
 }
}

/* calculate PML Hz10 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
```

```
    hzx[10][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[10][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
    *(ey[i+1][pml_ydim+ydim+j][pml_zdim+k]-ey[i][pml_ydim+ydim+j][pml_zdim+k]);
    hzy[10][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[10][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ex[i][pml_ydim+ydim+j+1][pml_zdim+k]-ex[i][pml_ydim+ydim+j][pml_zdim+k]);
    hz[i][pml_ydim+ydim+j][pml_zdim+k] = hzx[10][i][j][k] + hzy[10][i][j][k];
   }
  }
 }

 /* calculate PML Hz11 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hzx[11][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[11][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
    *(ey[pml_xdim+xdim+i+1][pml_ydim+ydim+j][pml_zdim+k]-ey[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
    hzy[11][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[11][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ex[pml_xdim+xdim+i][pml_ydim+ydim+j+1][pml_zdim+k]-ex[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
    hz[pml_xdim+xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hzx[11][i][j][k] + hzy[11][i][j][k];
   }
  }
 }

 /* calculate PML Hz12 *//*●*/
 for(i=0;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[12][i][j][k] = hzx[12][i][j][k]
    - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][j][k]-ey[pml_xdim+i][j][k]);
    hzy[12][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[12][i][j][k]
    + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[pml_xdim+i][j+1][k]-ex[pml_xdim+i][j][k]);
    hz[pml_xdim+i][j][k] = hzx[12][i][j][k] + hzy[12][i][j][k];
   }
  }
 }

 /* calculate PML Hz13 *//*●*/
 for(i=0;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[13][i][j][k] = hzx[13][i][j][k]
    - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][j][pml_zdim+zdim+k]-ey[pml_xdim+i][j][pml_zdim+zdim+k]);
    hzy[13][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[13][i][j][k]
    + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)
    *(ex[pml_xdim+i][j+1][pml_zdim+zdim+k]-ex[pml_xdim+i][j][pml_zdim+zdim+k]);
    hz[pml_xdim+i][j][pml_zdim+zdim+k] = hzx[13][i][j][k] + hzy[13][i][j][k];
   }
  }
 }

 /* calculate PML Hz14 *//*●*/
 for(i=0;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[14][i][j][k] = hzx[14][i][j][k]
    - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][pml_ydim+ydim+j][k]-ey[pml_xdim+i][pml_ydim+ydim+j][k]);
    hzy[14][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[14][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)*(ex[pml_xdim+i][pml_ydim+ydim+j+1][k]-ex[pml_xdim+i][pml_ydim+ydim+j][k]);
    hz[pml_xdim+i][pml_ydim+ydim+j][k] = hzx[14][i][j][k] + hzy[14][i][j][k];
   }
  }
 }

 /* calculate PML Hz15 *//*●*/
 for(i=0;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[15][i][j][k] = hzx[15][i][j][k] - (dt/m)*(1.0/dx)
```

```
    *(ey[pml_xdim+i+1][pml_ydim+ydim+j][pml_zdim+zdim+k]-ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hzy[15][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[15][i][j][k]
    + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
    *(ex[pml_xdim+i][pml_ydim+ydim+j+1][pml_zdim+zdim+k]-ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k]);
    hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+zdim+k] = hzx[15][i][j][k] + hzy[15][i][j][k];
   }
  }
 }

 /* calculate PML Hz16 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[16][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[16][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][pml_ydim+j][k]-ey[i][pml_ydim+j][k]);
    hzy[16][i][j][k] = hzy[16][i][j][k]
    + (dt/m)*(1.0/dy)*(ex[i][pml_ydim+j+1][k]-ex[i][pml_ydim+j][k]);
    hz[i][pml_ydim+j][k] = hzx[16][i][j][k] + hzy[16][i][j][k];
   }
  }
 }

 /* calculate PML Hz17 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<ydim;j++){
   for(k=1;k<=pml_zdim;k++){
    hzx[17][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[17][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)*(ey[pml_xdim+xdim+i+1][pml_ydim+j][k]-ey[pml_xdim+xdim+i][pml_ydim+j][k]);
    hzy[17][i][j][k] = hzy[17][i][j][k]
    + (dt/m)*(1.0/dy)*(ex[pml_xdim+xdim+i][pml_ydim+j+1][k]-ex[pml_xdim+xdim+i][pml_ydim+j][k]);
    hz[pml_xdim+xdim+i][pml_ydim+j][k] = hzx[17][i][j][k] + hzy[17][i][j][k];
   }
  }
 }

 /* calculate PML Hz18 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[18][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[18][i][j][k]
    - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)
    *(ey[i+1][pml_ydim+j][pml_zdim+zdim+k]-ey[i][pml_ydim+j][pml_zdim+zdim+k]);
    hzy[18][i][j][k] = hzy[18][i][j][k]
    + (dt/m)*(1.0/dy)*(ex[i][pml_ydim+j+1][pml_zdim+zdim+k]-ex[i][pml_ydim+j][pml_zdim+zdim+k]);
    hz[i][pml_ydim+j][pml_zdim+zdim+k] = hzx[18][i][j][k] + hzy[18][i][j][k];
   }
  }
 }

 /* calculate PML Hz19 *//*●*/
 for(i=0;i<pml_xdim;i++){
  for(j=0;j<ydim;j++){
   for(k=0;k<pml_zdim;k++){
    hzx[19][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[19][i][j][k]
    - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
    *(ey[pml_xdim+xdim+i+1][pml_ydim+j][pml_zdim+zdim+k]-ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
    hzy[19][i][j][k] = hzy[19][i][j][k] + (dt/m)*(1.0/dy)
    *(ex[pml_xdim+xdim+i][pml_ydim+j+1][pml_zdim+zdim+k]-ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
    hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hzx[19][i][j][k] + hzy[19][i][j][k];
   }
  }
 }

 /* calculate PML Hz20 *//*●*/
 for(i=0;i<xdim;i++){
  for(j=0;j<pml_ydim;j++){
   for(k=1;k<zdim;k++){
    hzx[20][i][j][k] = hzx[20][i][j][k]
    - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][j][pml_zdim+k]-ey[pml_xdim+i][j][pml_zdim+k]);
    hzy[20][i][j][k] = (1.0-(roy[pml_ydim-j-1]*dt)/(2.0*e))/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*hzy[20][i][j][k]
```

```
   + (dt/m)/(1.0+(roy[pml_ydim-j-1]*dt)/(2.0*e))*(1.0/dy)*(ex[pml_xdim+i][j+1][pml_zdim+k]-ex[pml_xdim+i][j][pml_zdim+k]);
   hz[pml_xdim+i][j][pml_zdim+k] = hzx[20][i][j][k] + hzy[20][i][j][k];
  }
 }
}

/* calculate PML Hz21 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<pml_ydim;j++){
  for(k=1;k<zdim;k++){
   hzx[21][i][j][k] = hzx[21][i][j][k]
   - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][pml_ydim+ydim+j][pml_zdim+k]-ey[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hzy[21][i][j][k] = (1.0-(roy[j]*dt)/(2.0*e))/(1.0+(roy[j]*dt)/(2.0*e))*hzy[21][i][j][k]
   + (dt/m)/(1.0+(roy[j]*dt)/(2.0*e))*(1.0/dy)
   *(ex[pml_xdim+i][pml_ydim+ydim+j+1][pml_zdim+k]-ex[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k]);
   hz[pml_xdim+i][pml_ydim+ydim+j][pml_zdim+k] = hzx[21][i][j][k] + hzy[21][i][j][k];
  }
 }
}

/* calculate PML Hz22 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hzx[22][i][j][k] = (1.0-(rox[pml_xdim-i-1]*dt)/(2.0*e))/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*hzx[22][i][j][k]
   - (dt/m)/(1.0+(rox[pml_xdim-i-1]*dt)/(2.0*e))*(1.0/dx)*(ey[i+1][pml_ydim+j][pml_zdim+k]-ey[i][pml_ydim+j][pml_zdim+k]);
   hzy[22][i][j][k] = hzy[22][i][j][k]
   + (dt/m)*(1.0/dy)*(ex[i][pml_ydim+j+1][pml_zdim+k]-ex[i][pml_ydim+j][pml_zdim+k]);
   hz[i][pml_ydim+j][pml_zdim+k] = hzx[22][i][j][k] + hzy[22][i][j][k];
  }
 }
}

/* calculate PML Hz23 *//*●*/
for(i=0;i<pml_xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<zdim;k++){
   hzx[23][i][j][k] = (1.0-(rox[i]*dt)/(2.0*e))/(1.0+(rox[i]*dt)/(2.0*e))*hzx[23][i][j][k]
   - (dt/m)/(1.0+(rox[i]*dt)/(2.0*e))*(1.0/dx)
   *(ey[pml_xdim+xdim+i+1][pml_ydim+j][pml_zdim+k]-ey[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hzy[23][i][j][k] = hzy[23][i][j][k]
   + (dt/m)*(1.0/dy)*(ex[pml_xdim+xdim+i][pml_ydim+j+1][pml_zdim+k]-ex[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k]);
   hz[pml_xdim+xdim+i][pml_ydim+j][pml_zdim+k] = hzx[23][i][j][k] + hzy[23][i][j][k];
  }
 }
}

/* calculate PML Hz24 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=1;k<=pml_zdim;k++){
   hzx[24][i][j][k] = hzx[24][i][j][k]
   - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][pml_ydim+j][k]-ey[pml_xdim+i][pml_ydim+j][k]);
   hzy[24][i][j][k] = hzy[24][i][j][k]
   + (dt/m)*(1.0/dy)*(ex[pml_xdim+i][pml_ydim+j+1][k]-ex[pml_xdim+i][pml_ydim+j][k]);
   hz[pml_xdim+i][pml_ydim+j][k] = hzx[24][i][j][k] + hzy[24][i][j][k];
  }
 }
}

/* calculate PML Hz25 *//*●*/
for(i=0;i<xdim;i++){
 for(j=0;j<ydim;j++){
  for(k=0;k<pml_zdim;k++){
   hzx[25][i][j][k] = hzx[25][i][j][k]
   - (dt/m)*(1.0/dx)*(ey[pml_xdim+i+1][pml_ydim+j][pml_zdim+zdim+k]-ey[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hzy[25][i][j][k] = hzy[25][i][j][k]
   + (dt/m)*(1.0/dy)*(ex[pml_xdim+i][pml_ydim+j+1][pml_zdim+zdim+k]-ex[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k]);
   hz[pml_xdim+i][pml_ydim+j][pml_zdim+zdim+k] = hzx[25][i][j][k] + hzy[25][i][j][k];
  }
```

```
 }
}

/* ■■■■■■■■■■■■■■■■■■ PML for H end ■■■■■■■■■■■■■■■■■■ */

if(count%5 == 0){

 sprintf(fname,"ez_data3_t%d.txt",fcount);
 output_fp = fopen(fname, "a");
 if(output_fp == NULL){
  printf("ファイルが開けません\n");
  exit(-1);
 }

 for(j=pml_ydim+1;j<pml_ydim+ydim;j++){
  for(i=pml_xdim+1;i<pml_xdim+xdim;i++){
   fprintf(output_fp, "%lf,", ez[i][j][pml_zdim+f_z[0]]);
  }
  fprintf(output_fp, "\n");
 }
 fclose(output_fp);
 fcount++;

 /* PPMデータ作成 */
 if(ppm_flg == 1){
  char command[256] = "make_gif_from_ppm.bat ";

  make_ppm_p6(fname, 100);
  strcat(command, fname);
  system(command);
 }
}

/* 観測点でのデータ収集 */
if(observe_flg == 1){
 fprintf(output_observe, "%16.15lf,%16.15lf\n", t, ez[pml_xdim+x_obse][pml_ydim+y_obse][pml_zdim+z_obse]);
}

printf("現在全体の [%lf]%の処理が終わっています\r",((t/tmax)*100.0));
count++;
}

/* 計算所要時間計測終了点 */
time_data = end_clc_time();

/* 計算所要時間表示 */
fprintf(log_fp, "%s\n", time_data);

/* ------------------------------------------------------------------------------------------
ID配列の開放
------------------------------------------------------------------------------------------ */

free(ID_data_struct.eps);
free(ID_data_struct.mu);
free(ID_data_struct.ro);
/* ID_data_struct自身は自動変数なので開放の必要なし */

/* ------------------------------------------------------------------------------------------
周波数周りの配列f_x,f_y,f_z,fの開放
------------------------------------------------------------------------------------------ */

free(f_x);
free(f_y);
free(f_z);
free(f);

/* ------------------------------------------------------------------------------------------
減衰項配列rox[],roy[],roz[]の開放
------------------------------------------------------------------------------------------ */
```

```
 free(rox);
 free(roy);
 free(roz);

 /* ----------------------------------------------------------------------------------------------
 計算用配列の開放
 ---------------------------------------------------------------------------------------------- */

 free_3Darray(ex, xdim + 2*pml_xdim, ydim + 2*pml_ydim);
 free_3Darray(ey, xdim + 2*pml_xdim, ydim + 2*pml_ydim);
 free_3Darray(ez, xdim + 2*pml_xdim, ydim + 2*pml_ydim);
 free_3Darray(hx, xdim + 2*pml_xdim, ydim + 2*pml_ydim);
 free_3Darray(hy, xdim + 2*pml_xdim, ydim + 2*pml_ydim);
 free_3Darray(hz, xdim + 2*pml_xdim, ydim + 2*pml_ydim);

 free_3Darray_PML(exy, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(exz, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(eyx, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(eyz, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(ezx, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(ezy, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hxy, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hxz, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hyx, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hyz, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hzx, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);
 free_3Darray_PML(hzy, pml_xdim, pml_ydim, pml_zdim, xdim, ydim, zdim);

 /* ----------------------------------------------------------------------------------------------
 ファイルポインタのクローズ
 ---------------------------------------------------------------------------------------------- */

 fclose(log_fp);
 fclose(input_file_fp);
 fclose(input_opengl_fp);
 if(observe_flg == 1){
  fclose(output_observe);
 }

 return(1);

}
```

●●● check_input.h ●●●

```
#ifndef _CHECK_INPUT_H_INCLUED_
#define _CHECK_INPUT_H_INCLUED_

struct input_data{ //入力データ数が増えたらここに記述
 int f_num;
 int M_x, M_y, M_z;
 int dB_x, dB_y, dB_z;
 int *f_x, *f_y, *f_z;
 int xdim, ydim, zdim;
 int pml_xdim, pml_ydim, pml_zdim;
 int boundary_flg;
 int ppm_flg, observe_flg;
 int x_obse, y_obse, z_obse;
 double tmax;
 double *f;
 double dx,dy,dz,dt,t;
 double *f_start, *f_end;
 double clc_cycle;
 char output_file_name[256];
};

struct ID_data_for_setting{
 int ID_num;
 int *ID_i_start;
 int *ID_i_end;
```

```
 int *ID_j_start;
 int *ID_j_end;
 int *ID_k_start;
 int *ID_k_end;
};

struct ID_data{
 double *eps;
 double *mu;
 double *ro;
};

/* 入力データの整合性を確かめる関数 */
/* 安定条件を満たさないならば-1(ERROR) を返す */
int check_input(input_data *, ID_data_for_setting *);

#endif _CHECK_INPUT_H_INCLUED_
```

●●● check_input.cpp ●●●

```
#include<math.h>
#include<stdio.h>
#include"check_input.h"
#include"debug.h"

/* ------------------------------------------------------------------------------------------------
セルサイズを決定する関数
引数である周波数での波長，それの 10 分の 1 とする
詳しくは文献 (FDTD 法による電磁界およびアンテナ解析：宇野亨，pp50) 参照
------------------------------------------------------------------------------------------------ */

static double determine_dxyz(double f){

 double lamda, dx;
 lamda = (3.0*(pow(10.0, 8.0)))/f;
 dx = lamda / 10.0; // 安定化のために，セルサイズは波長の 1/10
 LOG((stderr, "dx = %lf -- %s -- %d\n", dx, __FILE__, __LINE__));
 return(dx);
}

/* ------------------------------------------------------------------------------------------------
計算最大時間を決定する
引数は，セルサイズ
計算を安定させるために，さらに自身の 100 分の 1 だけ小さくする
『Courant の安定条件』
詳しくは文献 (FDTD 法による電磁界およびアンテナ解析：宇野亨，pp52) 参照
------------------------------------------------------------------------------------------------ */

static double determine_dt(double dx){

 double dt;
 dt = dx/(sqrt(3.0)*3.0*(pow(10.0, 8.0)));
 dt -= dt/100.0; // 計算安定化のため時間ステップを 1%分小さめに
 LOG((stderr, "dt = %lf -- %s -- %d\n", dt, __FILE__, __LINE__));
 return(dt);
}

/* ------------------------------------------------------------------------------------------------
計算最大時間を決定する
引数は，時間ステップサイズ
時間ステップは入力周波数の約 10 倍から計算されている.dt は，入力周波数の周期の約 1/10
------------------------------------------------------------------------------------------------ */

static double determine_tmax(double dt, double clc_cycle){

 double tmax;
 tmax = clc_cycle*10.0*dt; // 10 倍の周波数で時間ステップ (dt) を規定しているので
 LOG((stderr, "tmax = %lf -- %s -- %d\n", tmax, __FILE__, __LINE__));
 return(tmax);
}
```

```
/* ------------------------------------------------------------------------------------------
ローカル関数を呼び出して，入力データが安定条件を満たしているかを確認
------------------------------------------------------------------------------------------ */
int check_input(input_data *struct_data, ID_data_for_setting *ID_data_for_setting_data){

 int i;
 double fmax=0.0, f_tmp;

 /* 時間ステップの計算（これは読み込むには面倒なので関数で実現） */
 struct_data->dt = determine_dt(struct_data->dx);
 LOG((stderr, "struct_data->dt = %16.15lf\n", struct_data->dt));

 /* 入力データの中から最大周波数を検出 */
 for(i=0;i<struct_data->f_num;i++){
  f_tmp = struct_data->f[i];
  if(struct_data->f[i] >= f_tmp && struct_data->f[i] >= fmax){
   fmax = struct_data->f[i];
  }
 }

 /* dx, dy, dz の安定条件に関してチェック */
 if(struct_data->dx > determine_dxyz(fmax)){
  printf("入力データ (dx) が安定条件を満たしません\n");
  return(ERROR);
 }
 if(struct_data->dy > determine_dxyz(fmax)){
  printf("入力データ (dy) が安定条件を満たしません\n");
  return(ERROR);
 }
 if(struct_data->dz > determine_dxyz(fmax)){
  printf("入力データ (dz) が安定条件を満たしません\n");
  return(ERROR);
 }

 /* 座標の整合性について検証 */
 for(i=1;i<=ID_data_for_setting_data->ID_num;i++){
  /* ID の位置と解析空間配列数の関係 */
  if(0 > ID_data_for_setting_data->ID_i_start[i] || struct_data->xdim < ID_data_for_setting_data->ID_i_end[i]){
   printf("ID の座標 [i] の位置が解析空間外です\n");
   return(ERROR);
  }
  if(0 > ID_data_for_setting_data->ID_j_start[i] || struct_data->ydim < ID_data_for_setting_data->ID_j_end[i]){
   printf("ID の座標 [j] の位置が解析空間外です\n");
   return(ERROR);
  }
  if(0 > ID_data_for_setting_data->ID_k_start[i] || struct_data->zdim < ID_data_for_setting_data->ID_k_end[i]){
   printf("ID の座標 [k] の位置が解析空間外です\n");
   return(ERROR);
  }
  /* ID の位置の大小関係 */
  if(ID_data_for_setting_data->ID_i_start[i] > ID_data_for_setting_data->ID_i_end[i]){
   printf("ID の座標 [i] の大小関係が矛盾しています\n");
   return(ERROR);
  }
  if(ID_data_for_setting_data->ID_j_start[i] > ID_data_for_setting_data->ID_j_end[i]){
   printf("ID の座標 [j] の大小関係が矛盾しています\n");
   return(ERROR);
  }
  if(ID_data_for_setting_data->ID_k_start[i] > ID_data_for_setting_data->ID_k_end[i]){
   printf("ID の座標 [k] の大小関係が矛盾しています\n");
   return(ERROR);
  }
 }

 /* 信号座標と解析空間サイズ */
 for(i=0;i<struct_data->f_num;i++){
  if(struct_data->f_x[i] > struct_data->xdim || struct_data->f_x[i] < 0){
   printf("f_x が解析空間外です\n");
```

```
   return(ERROR);
  }
  if(struct_data->f_y[i] > struct_data->ydim || struct_data->f_y[i] < 0){
   printf("f_y が解析空間外です\n");
   return(ERROR);
  }
  if(struct_data->f_z[i] > struct_data->zdim || struct_data->f_z[i] < 0){
   printf("f_z が解析空間外です\n");
   return(ERROR);
  }
 }

 /* 観測点の妥当性（電界や磁界もしくはその各成分によって解析空間の大きさはことなるので，解析空間外（境界面を含む）はエラーとした）*/
 if(struct_data->x_obse <= 0 || struct_data->x_obse >= struct_data->xdim){
  printf("観測点 x 座標が解析空間外です\n");
  return(ERROR);
 }
 if(struct_data->y_obse <= 0 || struct_data->y_obse >= struct_data->ydim){
  printf("観測点 y 座標が解析空間外です\n");
  return(ERROR);
 }
 if(struct_data->z_obse <= 0 || struct_data->z_obse >= struct_data->zdim){
  printf("観測点 z 座標が解析空間外です\n");
  return(ERROR);
 }

 /* tmax を決定する */
 struct_data->tmax = determine_tmax(struct_data->dt, struct_data->clc_cycle);
 LOG((stderr, "tmax = %16.15lf\n", struct_data->tmax));

 return(SUCCESS);

}
```

●●● clc_time.h ●●●

```
#ifndef _CLC_TIME_H_INCLUED_
#define _CLC_TIME_H_INCLUED_

/* ■モジュールの使用法■
 * 1.char* の変数を呼び出し側で作成
 * 2. 次の関数をコール start_clc_time();
 * 3. 次の関数をコール ---> 戻りを 1. で作成した変数を入れる char* = end_clc_time();
 * 4. 戻り値を呼び出し側で使用する 【例】printf("%s\n", char*);
 */


/* 測定したい時間の開始点を設定する
 * 引数：void
 * 戻り値：void
 */
void start_clc_time(void);

/* 測定したい時間の終点を設定し計算時間を返す
 * 引数：void
 * 戻り値：char*(経過時間を下記のフォーマットで返す)
 * "計算所要時間 : %d 日%d 時間%d 分%d 秒"
 */
char *end_clc_time(void);

#endif /* _CLC_TIME_H_INCLUED_ */
```

●●● clc_time.cpp ●●●

```
#include<time.h>
#include"clc_time.h"
#include<stdio.h>

static long time_start, time_end;
```

```
static char *clc_time(void){

 char tmp[256];
 int day, hour, min, sec;
 long clc_time;

 clc_time = time_end - time_start;

 sec = clc_time%60;
 min = ((clc_time-sec)/60)%60;
 hour = ((clc_time-min*60-sec)/60/60)%24;
 day = ((clc_time-hour*60*60-min*60-sec)/60/60/24)%365;

 sprintf(tmp, "計算所要時間 : %d 日%d 時間%d 分%d 秒", day, hour, min, sec);
 return(tmp);
}

void start_clc_time(void){
 time_start = time(NULL);
}

char *end_clc_time(void){
 time_end = time(NULL);
 return(clc_time());
}
```

●●● debug.h ●●●

```
#ifndef _DEBUG_H_INCLUED_
#define _DEBUG_H_INCLUED_

#define NDBUG /* この行をコメントにするとデバックモード */

#define SUCCESS 1
#define MALLOC_ERROR -1
#define FGETS_ERROR -2
#define ARGUMENT_ERROR -3
#define FORMAT_NOT_FIT_ARGUMENT_ERROR -4
#define FILE_OPEN_ERROR -5
#define ERROR -6

#ifdef NDBUG
#define LOG(args) do{/* none */}while(0)
#else
#define LOG(args) fprintf args
#endif

/* ------------------------------------------------------------------------
・デバッグモードのみで出力したい行は関数内で下記のように記述してください
LOG((stderr, "hello world -- %s -- %d\n", __FILE__, __LINE__));
------------------------------------------------------------------------ */

#endif /* _DEBUG_H_INCLUED_ */
```

●●● input_data.h ●●●

```
#ifndef _INPUT_DATA_H_INCLUED_
#define _INPUT_DATA_H_INCLUED_

#include<stdio.h>
#include"check_input.h"

/* ------------------------------------------------------------------------------------------------------------------------
 ■ [input_data_from_file] 概要
 設定ファイルから，データを読み込み，引数で受けたデータ構造体にデータを代入する

 ■引数
 ・読み込みたい設定ファイルへのポインタ，入力データを代入したい変数のポインタ
 ■戻り値
```

```
 ・読み込みに成功した時 1 を返し，エラーの場合は-1 を返す

 ■注意
 ・構造体のメンバ,f, f_x, f_y, f_z, f_start, f_end は malloc で確保しています ---> free の必要あり
------------------------------------------------------------------------------------------------------------------------------------------------------ */
int input_data_from_file(FILE *, input_data *);

/* ------------------------------------------------------------------------------------------------------------------------------------------------------
 ■ [set_ID_data] 概要
 設定ファイルから，データを読み込み，引数で受けたデータ構造体にデータを代入する

 ■引数
 ・読み込みたい設定ファイルへのポインタ，入力データを代入したい変数のポインタ，構造体 [ID_data]
 ■戻り値
 ・読み込みに成功した時 1 を返し，エラーの場合は-1 を返す

 ■注意
 ・構造体のメンバ,ID_i_start,ID_i_end,ID_j_start,ID_j_end,ID_k_start,ID_k_end は malloc で確保しています ---> free
の必要あり
------------------------------------------------------------------------------------------------------------------------------------------------------ */
int set_ID_data(FILE *, ID_data_for_setting *, ID_data *);

#endif /* _INPUT_DATA_H_INCLUED_ */
```

●●● input_data.cpp ●●●

```
#include<math.h>
#include<stdlib.h>
#include"input_data.h"
#include"check_input.h"
#include"xsscanf.h"
#include"debug.h"

int input_data_from_file(FILE *inputfile_fp, input_data *struct_data){

 int i;

 /* 入力信号数読み込み */
 xsscanf(inputfile_fp, "%d", &struct_data->f_num);
 LOG((stderr, "struct_data->f_num = %d\n", struct_data->f_num));

 /* 入力信号数周波数読み込み */
 struct_data->f = (double*)malloc(sizeof(double)*struct_data->f_num);
 if(struct_data->f == NULL){
  LOG((stderr, "struct_data->f malloc error\n"));
  return(ERROR);
 }
 for(i=0;i<struct_data->f_num;i++){
  xsscanf(inputfile_fp, "%lf", &struct_data->f[i]);
  LOG((stderr, "struct_data->f = %lf\n", struct_data->f[i]));
 }

 /* 入力信号の励振源（位置）読み込み */
 struct_data->f_x = (int*)malloc(sizeof(int)*struct_data->f_num);
 if(struct_data->f_x == NULL){
  printf("struct_data->f_x malloc error\n");
  return(ERROR);
 }
 struct_data->f_y = (int*)malloc(sizeof(int)*struct_data->f_num);
 if(struct_data->f_y == NULL){
  printf("struct_data->f_y malloc error\n");
  return(ERROR);
 }
 struct_data->f_z = (int*)malloc(sizeof(int)*struct_data->f_num);
 if(struct_data->f_z == NULL){
  printf("struct_data->f_z malloc error\n");
  return(ERROR);
 }
 for(i=0;i<struct_data->f_num;i++){
  xsscanf(inputfile_fp, "%d %d %d", &struct_data->f_x[i], &struct_data->f_y[i], &struct_data->f_z[i]);
```

```
 LOG((stderr, "struct_data->f_x = %d struct_data->f_y = %d struct_data->f_z = %d\n",
      struct_data->f_x[i], struct_data->f_y[i], struct_data->f_z[i]));
}

/* 入力信号の継続期間読み込み */
struct_data->f_start = (double*)malloc(sizeof(double)*struct_data->f_num);
if(struct_data->f_start == NULL){
 printf("struct_data->f_start malloc error\n");
 return(ERROR);
}
struct_data->f_end = (double*)malloc(sizeof(double)*struct_data->f_num);
if(struct_data->f_end == NULL){
 printf("struct_data->f_end malloc error\n");
 return(ERROR);
}
for(i=0;i<struct_data->f_num;i++){
 xsscanf(inputfile_fp, "%lf %lf", &struct_data->f_start[i], &struct_data->f_end[i]);
 LOG((stderr, "struct_data->f_start = %lf struct_data->f_end = %lf\n",
      struct_data->f_start[i], struct_data->f_end[i]));
}

/* 計算周期読み込み */
xsscanf(inputfile_fp, "%lf", &struct_data->clc_cycle);
LOG((stderr, "struct_data->clc_cycle = %lf\n", struct_data->clc_cycle));

/* セル数読み込み */
xsscanf(inputfile_fp, "%d %d %d", &struct_data->xdim, &struct_data->ydim, &struct_data->zdim);
LOG((stderr, "struct_data->xdim = %d struct_data->ydim = %d struct_data->zdim = %d\n",
     struct_data->xdim, struct_data->ydim, struct_data->zdim));

/* セルサイズ読み込み */
xsscanf(inputfile_fp, "%lf %lf %lf", &struct_data->dx, &struct_data->dy, &struct_data->dz);
LOG((stderr, "struct_data->dx = %lf struct_data->dy = %lf struct_data->dz = %lf\n",
     struct_data->dx, struct_data->dy, struct_data->dz));

/* 境界条件読み込み */
xsscanf(inputfile_fp, "%d", &struct_data->boundary_flg);
LOG((stderr, "struct_data->boundary_flg = %d\n", struct_data->boundary_flg));

/* PML層の層数読み込み */
xsscanf(inputfile_fp, "%d %d %d", &struct_data->pml_xdim, &struct_data->pml_ydim, &struct_data->pml_zdim);
LOG((stderr, "struct_data->pml_xdim = %d struct_data->pml_ydim = %d struct_data->pml_zdim = %d\n",
     struct_data->pml_xdim, struct_data->pml_ydim, struct_data->pml_zdim));

/* PML層の減衰項 (M) 読み込み */
xsscanf(inputfile_fp, "%d %d %d", &struct_data->M_x, &struct_data->M_y, &struct_data->M_z);
LOG((stderr, "struct_data->M_x = %d struct_data->M_y = %d struct_data->M_z = %d\n",
     struct_data->M_x, struct_data->M_y, struct_data->M_z));

/* PML層の減衰項 (dB) 読み込み */
xsscanf(inputfile_fp, "%d %d %d", &struct_data->dB_x, &struct_data->dB_y, &struct_data->dB_z);
LOG((stderr, "struct_data->dB_x = %d struct_data->dB_y = %d struct_data->dB_z = %d\n",
     struct_data->dB_x, struct_data->dB_y, struct_data->dB_z));

/* 計算データ出力ファイル名読み込み */
xsscanf(inputfile_fp, "%s", struct_data->output_file_name);
LOG((stderr, "struct_data->output_file_name = %s\n", struct_data->output_file_name));

/* PPM計算の実行の有無読み込み */
xsscanf(inputfile_fp, "%d", &struct_data->ppm_flg);
LOG((stderr, "struct_data->ppm_flg = %d\n", struct_data->ppm_flg));

/* 観測点でのデータ収集の実行の有無読み込み */
xsscanf(inputfile_fp, "%d", &struct_data->observe_flg);
LOG((stderr, "struct_data->observe_flg = %d\n", struct_data->observe_flg));

/* 観測点の座標読み込み */
if(struct_data->observe_flg == 1){/* 観測あり:1 観測なし:0 */
 xsscanf(inputfile_fp, "%d %d %d", &struct_data->x_obse, &struct_data->y_obse, &struct_data->z_obse);
 LOG((stderr, "struct_data->x_obse = %d struct_data->y_obse = %d struct_data->z_obse = %d\n",
```

```
        struct_data->x_obse, struct_data->y_obse, struct_data->z_obse));
  }

  return(SUCCESS);
}

/* ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ */

int set_ID_data(FILE *fp, ID_data_for_setting *ID_data_for_setting_data, ID_data *ID_data_struct){

  int i;
  double e=8.854187817*(pow(10.0, -12.0)), m=1.256637061*(pow(10.0, -6.0)), r=0.0; // 誘電率,透磁率,導電率の初
期値

  /* 入力材質数読み込み */
  xsscanf(fp, "%d", &ID_data_for_setting_data->ID_num);
  LOG((stderr, "ID_data_for_setting_data->ID_num = %d\n", ID_data_for_setting_data->ID_num));

  /* malloc ID_data_for_setting_data */
  ID_data_for_setting_data->ID_i_start = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_i_start == NULL){
   printf("ID_i_start malloc error\n");
   return(ERROR);
  }
  ID_data_for_setting_data->ID_i_end = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_i_end == NULL){
   printf("ID_i_end malloc error\n");
   return(ERROR);
  }
  ID_data_for_setting_data->ID_j_start = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_j_start == NULL){
   printf("ID_j_start malloc error\n");
   return(ERROR);
  }
  ID_data_for_setting_data->ID_j_end = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_j_end == NULL){
   printf("ID_j_end malloc error\n");
   return(ERROR);
  }
  ID_data_for_setting_data->ID_k_start = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_k_start == NULL){
   printf("ID_k_start malloc error\n");
   return(ERROR);
  }
  ID_data_for_setting_data->ID_k_end = (int *)malloc(sizeof(int)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_for_setting_data->ID_k_end == NULL){
   printf("ID_k_end malloc error\n");
   return(ERROR);
  }

  /* malloc ID_data_struct */
  ID_data_struct->eps = (double *)malloc(sizeof(double)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_struct->eps == NULL){
   printf("eps malloc error\n");
   return(ERROR);
  }
  ID_data_struct->mu = (double *)malloc(sizeof(double)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_struct->mu == NULL){
   printf("mu malloc error\n");
   return(ERROR);
  }
  ID_data_struct->ro = (double *)malloc(sizeof(double)*(ID_data_for_setting_data->ID_num+1));
  if(ID_data_struct->ro == NULL){
   printf("ro malloc error\n");
   return(ERROR);
  }

  /* 材料の座標代入 */
  for(i=1;i<=ID_data_for_setting_data->ID_num;i++){
   xsscanf(fp, "%d %d %d %d %d %d",
```

```
        &ID_data_for_setting_data->ID_i_start[i], &ID_data_for_setting_data->ID_i_end[i],
        &ID_data_for_setting_data->ID_j_start[i], &ID_data_for_setting_data->ID_j_end[i],
        &ID_data_for_setting_data->ID_k_start[i], &ID_data_for_setting_data->ID_k_end[i]);
 LOG((stderr, "ID_i_start = %d ID_i_end = %d ID_j_start = %d ID_j_end = %d ID_k_start = %d ID_k_end = %d\n",
      ID_data_for_setting_data->ID_i_start[i], ID_data_for_setting_data->ID_i_end[i],
      ID_data_for_setting_data->ID_j_start[i], ID_data_for_setting_data->ID_j_end[i],
      ID_data_for_setting_data->ID_k_start[i], ID_data_for_setting_data->ID_k_end[i]));
}

/* 入力ファイルはセル番号1からに対して，配列は0からなので数値を書き換える */
for(i=1;i<=ID_data_for_setting_data->ID_num;i++){
 ID_data_for_setting_data->ID_i_start[i] -= 1;
 ID_data_for_setting_data->ID_i_end[i] -= 1;
 ID_data_for_setting_data->ID_j_start[i] -= 1;
 ID_data_for_setting_data->ID_j_end[i] -= 1;
 ID_data_for_setting_data->ID_k_start[i] -= 1;
 ID_data_for_setting_data->ID_k_end[i] -= 1;
}

/* 材料値の代入 */
ID_data_struct->eps[0] = e;
ID_data_struct->mu[0] = m;
ID_data_struct->ro[0] = r;
LOG((stderr, "eps = %16.15lf mu = %16.15lf ro = %16.15lf\n",
     ID_data_struct->eps[0], ID_data_struct->mu[0], ID_data_struct->ro[0]));
for(i=1;i<=ID_data_for_setting_data->ID_num;i++){
 xsscanf(fp, "%lf %lf %lf", &ID_data_struct->eps[i], &ID_data_struct->mu[i], &ID_data_struct->ro[i]);
 ID_data_struct->eps[i] = e*ID_data_struct->eps[i];
 ID_data_struct->mu[i] = m*ID_data_struct->mu[i];
 ID_data_struct->ro[i] = r*ID_data_struct->ro[i];
 LOG((stderr, "eps = %16.15lf mu = %16.15lf ro = %16.15lf\n",
      ID_data_struct->eps[i], ID_data_struct->mu[i], ID_data_struct->ro[i]));
}

return(SUCCESS);
}
```

●●● malloc_free.h ●●●

```
#ifndef _MALLOC_FREE_H_INCLUDE_
#define _MALLOC_FREE_H_INCLUDE_

#include"check_input.h"

/* ------------------------------------------------------------------------------------------------
3次元配列確保専用 malloc 関数
引数は x,y,z 方向の大きさ
malloc するサイズは，引数より1つサイズ
戻り値：double ***
------------------------------------------------------------------------------------------------ */

double ***malloc_3Darray(int, int, int);

/* ------------------------------------------------------------------------------------------------
3次元配列確保専用 malloc 関数
引数はファイルから持ってきた3次元の配列サイズを持っている構造体
malloc するサイズは，引数より1つサイズ
戻り値：double ****
------------------------------------------------------------------------------------------------ */
double ****malloc_3Darray_PML(input_data *);

/* ------------------------------------------------------------------------------------------------
3次元配列（材質 ID）確保専用 malloc 関数
引数はファイルから持ってきた3次元の配列サイズを持っている構造体
malloc するサイズは，引数より1つサイズ
戻り値：int ****
------------------------------------------------------------------------------------------------ */
int ***malloc_3Darray_for_ID(int, int, int);

/* ------------------------------------------------------------------------------------------------
```

```
3次元配列開放専用free関数
引数は,freeしたいポインタのポインタのポインタ,配列の1次元要素サイズ,2次元要素サイズ
3次元配列確保専用malloc関数と併用してください
---------------------------------------------------------------------------------------------- */
void free_3Darray(double***, int, int);

/* ----------------------------------------------------------------------------------------------
3次元配列開放専用free関数
引数は,freeしたいポインタのポインタのポインタのポインタ,3次元の配列サイズを持っている構造体
---------------------------------------------------------------------------------------------- */
void free_3Darray_PML(double ****, int, int, int, int, int, int);

#endif /*_MALLOC_FREE_H_INCLUDE_ */#ifndef _MALLOC_FREE_H_INCLUDE_
#define _MALLOC_FREE_H_INCLUDE_

#include"check_input.h"

/* ----------------------------------------------------------------------------------------------
3次元配列確保専用malloc関数
引数はx,y,z方向の大きさ
mallocするサイズは,引数より1つサイズ
戻り値:double ***
---------------------------------------------------------------------------------------------- */

double ***malloc_3Darray(int, int, int);

/* ----------------------------------------------------------------------------------------------
3次元配列確保専用malloc関数
引数はファイルから持ってきた3次元の配列サイズを持っている構造体
mallocするサイズは,引数より1つサイズ
戻り値:double ****
---------------------------------------------------------------------------------------------- */
double ****malloc_3Darray_PML(input_data *);

/* ----------------------------------------------------------------------------------------------
3次元配列(材質ID)確保専用malloc関数
引数はファイルから持ってきた3次元の配列サイズを持っている構造体
mallocするサイズは,引数より1つサイズ
戻り値:int ****
---------------------------------------------------------------------------------------------- */
int ***malloc_3Darray_for_ID(int, int, int);

/* ----------------------------------------------------------------------------------------------
3次元配列開放専用free関数
引数は,freeしたいポインタのポインタのポインタ,配列の1次元要素サイズ,2次元要素サイズ
3次元配列確保専用malloc関数と併用してください
---------------------------------------------------------------------------------------------- */
void free_3Darray(double***, int, int);

/* ----------------------------------------------------------------------------------------------
3次元配列開放専用free関数
引数は,freeしたいポインタのポインタのポインタのポインタ,3次元の配列サイズを持っている構造体
---------------------------------------------------------------------------------------------- */
void free_3Darray_PML(double ****, int, int, int, int, int, int);

#endif /*_MALLOC_FREE_H_INCLUDE_ */
```

●●● malloc_free.cpp ●●●

```
#include<stdlib.h>
#include<stdio.h>
#include"malloc_free.h"
#include"debug.h"

double ***malloc_3Darray(int x, int y, int z){

 int i,j,k;
 double ***p;

 p = (double ***)malloc(sizeof(double **)*(size_t)(x+1));
```

```
 if(p == NULL){
  free(p);
  printf("p first loop malloc error!");
  exit(0);
 }

 for(i=0;i<(x+1);i++){
  p[i] = (double **)malloc(sizeof(double *)*(size_t)(y+1));
  if(p[i] == NULL){
   while(i >= 0){
    free(p[i]);
    i--;
   }
   free(p);
   printf("p second loop malloc error!");
   exit(0);
  }
 }

 {
  int j_origin;

  for(i=0;i<(x+1);i++){
   for(j=0;j<(y+1);j++){
    p[i][j] = (double *)malloc(sizeof(double)*(size_t)(z+1));
    if(p[i][j] == NULL){
     j_origin = j;
     while(i >= 0){
      j = j_origin;
      while(j >= 0){
       free(p[i][j]);
       j--;
      }
      i--;
     }
     for(i=0;i<(x+1);i++){
      free(p[i]);
     }
     free(p);
     printf("p third malloc error!");
     exit(0);
    }
   }
  }
 }

 /* 初期化 */
 for(i=0;i<(x+1);i++){
  for(j=0;j<(y+1);j++){
   for(k=0;k<(z+1);k++){
    p[i][j][k] = 0.0;
   }
  }
 }

 return(p);
}

double ****malloc_3Darray_PML(input_data *data){

 int i;
 double ****p;

 p = (double****)malloc(sizeof(double***)*(size_t)26);
 if(p == NULL){
  printf("malloc error\n");
  return(NULL);
 }

 //malloc_3Darray を呼ぶ
```

```
 for(i=0;i<26;i++){
  if(i >= 0 && i <= 7){
   p[i] = malloc_3Darray(data->pml_xdim, data->pml_ydim, data->pml_zdim);
  }else if(i >= 8 && i <= 11){
   p[i] = malloc_3Darray(data->pml_xdim, data->pml_ydim, data->zdim);
  }else if(i >= 12 && i <= 15){
   p[i] = malloc_3Darray(data->xdim, data->pml_ydim, data->pml_zdim);
  }else if(i >= 16 && i <= 19){
   p[i] = malloc_3Darray(data->pml_xdim, data->ydim, data->pml_zdim);
  }else if(i >= 20 && i <= 21){
   p[i] = malloc_3Darray(data->xdim, data->pml_ydim, data->zdim);
  }else if(i >= 22 && i <= 23){
   p[i] = malloc_3Darray(data->pml_xdim, data->ydim, data->zdim);
  }else if(i >= 24 && i <= 25){
   p[i] = malloc_3Darray(data->xdim, data->ydim, data->pml_zdim);
  }
 }

 return(p);
}

int ***malloc_3Darray_for_ID(int x, int y, int z){

 int i,j,k;
 int ***p;

 p = (int ***)malloc(sizeof(int **)*(size_t)(x+1));
 if(p == NULL){
  free(p);
  printf("p first loop malloc error!");
  return(NULL);
 }

 for(i=0;i<(x+1);i++){
  p[i] = (int **)malloc(sizeof(int *)*(size_t)(y+1));
  if(p[i] == NULL){
   while(i >= 0){
    free(p[i]);
    i--;
   }
   free(p);
   printf("p second loop malloc error!");
   return(NULL);
  }
 }

 {
  int j_origin;

  for(i=0;i<(x+1);i++){
   for(j=0;j<(y+1);j++){
    p[i][j] = (int *)malloc(sizeof(int)*(size_t)(z+1));
    if(p[i][j] == NULL){
     j_origin = j;
     while(i >= 0){
      j = j_origin;
      while(j >= 0){
       free(p[i][j]);
       j--;
      }
      i--;
     }
     for(i=0;i<(x+1);i++){
      free(p[i]);
     }
     free(p);
     printf("p third malloc error!");
     return(NULL);
    }
   }
```

```
 }
}

/* 初期化 */
for(i=0;i<(x+1);i++){
 for(j=0;j<(y+1);j++){
  for(k=0;k<(z+1);k++){
   p[i][j][k] = 0;
  }
 }
}

return(p);
}

void free_3Darray(double ***p, int loop1, int loop2){

 int i, j;

 for(i=0;i<(loop1+1);i++){
  for(j=0;j<(loop2+1);j++){
   free(p[i][j]);
  }
 }
 for(i=0;i<(loop1+1);i++){
  free(p[i]);
 }
 free(p);
}

void free_3Darray_PML(double ****p, int pml_xdim, int pml_ydim, int pml_zdim, int xdim, int ydim, int zdim){

 int i;

 for(i=0;i<26;i++){
  if(i >= 0 && i <= 7){
   free_3Darray(p[i], pml_xdim, pml_ydim);
  }else if(i >= 8 && i <= 11){
   free_3Darray(p[i], pml_xdim, pml_ydim);
  }else if(i >= 12 && i <= 15){
   free_3Darray(p[i], xdim, pml_ydim);
  }else if(i >= 16 && i <= 19){
   free_3Darray(p[i], pml_xdim, ydim);
  }else if(i >= 20 && i <= 21){
   free_3Darray(p[i], xdim, pml_ydim);
  }else if(i >= 22 && i <= 23){
   free_3Darray(p[i], pml_xdim, ydim);
  }else if(i >= 24 && i <= 25){
   free_3Darray(p[i], xdim, ydim);
  }
 }
 free(p);
}
```

●●● set_ID.h ●●●

```
#ifndef _SET_ID_H_INCLUED_
#define _SET_ID_H_INCLUED_

#include<stdio.h>
#include"check_input.h"

int ***set_ID(input_data *, ID_data_for_setting *);

#endif _SET_ID_H_INCLUED_
```

●●● set_ID.cpp ●●●

```
#include<stdio.h>
#include<stdlib.h>
```

```
#include"set_ID.h"
#include"check_input.h"
#include"xsscanf.h"
#include"debug.h"
#include"check_input.h"
#include"malloc_free.h"

int ***set_ID(input_data *input_file_data, ID_data_for_setting *ID_data_for_setting_data){

 int ***ID;
 int i,j,k,l;

 /* ID配列の malloc */
 ID = malloc_3Darray_for_ID(input_file_data->xdim, input_file_data->ydim, input_file_data->zdim);
 if(ID == NULL){
  return(NULL);
 }

 /* ID配列の初期化 */
 for(i=0;i<=input_file_data->xdim;i++){
  for(j=0;j<=input_file_data->ydim;j++){
   for(k=0;k<=input_file_data->zdim;k++){
    ID[i][j][k] = 0;
   }
  }
 }

 /* ID配列にデータ入力 *//* 追加更新 H19/NOV/24 */
 for(l=1;l<=ID_data_for_setting_data->ID_num;l++){
  for(i=(ID_data_for_setting_data->ID_i_start[l]);i<=ID_data_for_setting_data->ID_i_end[l];i++){
   for(j=(ID_data_for_setting_data->ID_j_start[l]);j<=ID_data_for_setting_data->ID_j_end[l];j++){
    for(k=(ID_data_for_setting_data->ID_k_start[l]);k<=ID_data_for_setting_data->ID_k_end[l];k++){
     ID[i][j][k] = ID_data_for_setting_data->ID_num;
    }
   }
  }
 }

 /* 誘電率などを i,j,k を用いて呼び出しかた
 for(i=0;i<=input_file_data.xdim;i++){
  for(j=0;j<=input_file_data.ydim;j++){
   for(k=0;k<=input_file_data.zdim;k++){
    ID_data_struct.eps[(int)ID[i][j][k]];
    ID_data_struct.mu[(int)ID[i][j][k]];
    ID_data_struct.ro[(int)ID[i][j][k]];
   }
  }
 }
 */
 return(ID);
}
```

●●● xsscanf.h ●●●

```
#ifndef _XSCANF_H_INCLUED_
#define _XSCANF_H_INCLUED_

#include<stdio.h>
#include"debug.h"

/* ------------------------------------------------------------------------------------------------------------------------------
 ■概要
 この関数は,FDTDの設定ファイル読み込み専用プログラムです
 '#'をコメント行として読み飛ばし,設定データを書式に沿って読み込みます

 ■引数
 ・読み込みたい設定ファイルへのポインタ,設定ファイルから入力したい書式,読み込んだデータを格納する変数ポインタ(可変)

 ■戻り値
 ・読み込みに成功した時1を返し,エラーの場合は負の値を返す
```

```
 ・-1: malloc に失敗した時にこの値を返す
 ・-2: fgets に失敗した時 (fp が EOF を指しているのにさらに読み込もうとした時 etc) にこの値を返す
 ・-3: argument が不明のポインタを指しているときにこの値を返す
 ・-4: 引数 [format] と読み込みファイルの形式があっていない時にこの値を返す
 ・-5: そのほかのエラー (プログラム内部の問題)

 ■注意
 ・設定ファイルから入力したい書式は double 型,int 型,char*型 (文字列),char 型に対応しています
 ・設定ファイルの入力データのある行の途中に'#' があれば, それ以降はコメントとして扱います
 ・設定ファイルの入力データの区切り文字は, スペース, コンマ, タブの三つとします
 ・この関数を呼び出す側の書式と入力データの書式が異なる場合の動作は sscanf に依存
 ・この関数を呼び出す側の書式と入力データの個数が異なるときは-1 を返す
 ・可変引数の最大値は 256 と設定してあります
 ・入力データの最大データ長は 1024 バイト ---> それ以上の文字は無視される
 ・書式例 : "%lf %d %s %c"
------------------------------------------------------------------------------------------------------------------------ */
int xsscanf(FILE *fp, char *format, ...);

#endif /* _XSCANF_INCLUED_ */
```

●●● xsscanf.cpp ●●●

```
#include<stdio.h>
#include<stdarg.h>
#include<string.h>
#include<stdlib.h>
#include"xsscanf.h"

/* 考えられる状況
 * 1 : 先頭が'#', コメント行
 * 2 : データ行
 * 3 : 空行
 * 4 : 途中までスペースで改行
 * 5 : スペースが多いデータ列
 * 6 : 途中からコメント行
 * 7 : 入力データと書式の不一致
 */

union argument{
 double *data_double;
 int *data_int;
 char *str;
};

static int renew(char *str, char **buf){

 char *p;
 int i=0;

 LOG((stderr, "str in renew = %s\n", str));

 /* 引数 str を区切り文字" ,\t\n"で分解 ---> それぞれを buf に格納 */
 p = strtok(str, " ,\t\n");

 if(p == NULL){
  printf("空行を検出\n");
  return(ERROR);
 }

 LOG((stderr, "strlen(p) = %d\n", strlen(p)));
 buf[i] = (char*)malloc(sizeof(char)*(size_t)(strlen(p)+1));
 if(buf[i] == NULL){
  printf("malloc error\n");
  return(MALLOC_ERROR);
 }
 strncpy(buf[i], p, strlen(p));
 buf[i][strlen(p)] = NULL;
 LOG((stderr, "p = %s\n", p));
 while((p = strtok(NULL, " ,\t\n")) != NULL){
  if(p[0] == '#'){
```

```
   return(i);
  }
  i++;
  buf[i] = (char*)malloc(sizeof(char)*(size_t)(strlen(p)+1));
  if(buf[i] == NULL){
   printf("malloc error\n");
   return(MALLOC_ERROR);
  }
  LOG((stderr, "p = %s\n", p));
  strncpy(buf[i], p, strlen(p));
  buf[i][strlen(p)] = NULL;
 }
 return(i);
}

static argument *get_argument(va_list ap, char *format, int *arg_num, int *error_flg){

 int i, j=0;
 int str_len;
 argument *arg, *arg_tmp;

 str_len = strlen(format);

 arg = (argument *)malloc(sizeof(argument)*256);
 if(arg == NULL){
  *error_flg = 1;
  return(NULL);
 }

 for(i=0;i<str_len;i++){
  if(format[i] == '%'){
   if(format[i+1] == 'd'){
    arg[j].data_int = va_arg(ap, int*);
    #ifndef NDBUG
    if(arg[j].data_int == NULL){
     printf("第%d 可変引数が無効なポインタです\n", (j+1));
    }
    #endif
    j++;
   }else if(format[i+1] == 's'){
    arg[j].str = va_arg(ap, char*);
    #ifndef NDBUG
    if(arg[j].data_int == NULL){
     printf("第%d 可変引数が無効なポインタです\n", (j+1));
    }
    #endif
    j++;
   }else if(format[i+1] == 'c'){
    arg[j].str = va_arg(ap, char*);
    #ifndef NDBUG
    if(arg[j].data_int == NULL){
     printf("第%d 可変引数が無効なポインタです\n", (j+1));
    }
    #endif
    j++;
   }else if(format[i+1] == 'l' && format[i+2] == 'f'){
    arg[j].data_double = va_arg(ap, double*);
    #ifndef NDBUG
    if(arg[j].data_int == NULL){
     printf("第%d 可変引数が無効なポインタです\n", (j+1));
    }
    #endif
    j++;
   }else{
    *error_flg = 1;
   }
  }
 }
 *arg_num = j;
```

```
 return(arg);
}

int xsscanf(FILE *fp, char *format, ...){

 va_list ap;
 int flg=0;
 int arg_num;
 char buf[1024];
 char *buf_for_format;
 argument *argument1;
 char **tmp_format;
 char **tmp_buf;
 int tmp_format_num, tmp_buf_num;
 int k;

 #ifndef NDEBUG
 static int call_num = 1;
 LOG((stderr, "\n xsscanf -- %dth call\n\n", call_num));
 call_num++;
 #endif

 LOG((stderr, "format = %s\n", format));
 LOG((stderr, "strlen_format = %d\n", (int)strlen(format)));

 /* [buf_for_format] を作り引数 [format] をコピーする */
 /* 必要性の理由：[renew] 関数は引数の文字列を置換する (文字列が変質する) ため <--- ループをまわす時に問題が起きたため
修正 */
 buf_for_format = (char*)malloc(sizeof(char)*(strlen(format)+1));
 if(buf_for_format == NULL){
  printf("buf_for_format error\n");
  return(ERROR);
 }
 strncpy(buf_for_format, format, strlen(format));
 buf_for_format[(int)strlen(format)] = NULL;
 if((strncmp(buf_for_format, format, (strlen(format)+1))) != 0){
  printf("プログラム内部でエラーが起きました\n");
  return(ERROR);
 }

 /* 可変引数の取得 */
 va_start(ap, format);
 argument1 = get_argument(ap, format, &arg_num, &flg);
 va_end(ap);
 if(flg == 1){
  return(ARGUMENT_ERROR);
 }

 /* 共用体のアドレスに関して (参考までに) */
 /*
 printf("argument1[0] = %p\n", argument1[0]);
 printf("argument1[0].data_double = %p\n", argument1[0].data_double);
 */

 /* 引数 [format][buf] 用の配列を用意 */
 tmp_format = (char**)malloc(sizeof(char*)*256);
 if(tmp_format == NULL){
  printf("malloc error\n");
  return(MALLOC_ERROR);
 }
 tmp_buf = (char**)malloc(sizeof(char*)*256);
 if(tmp_buf == NULL){
  printf("malloc error\n");
  return(MALLOC_ERROR);
 }

 /* format 分解 */
 LOG((stderr, "buf_for_format in xsscanf = %s\n", buf_for_format));
 tmp_format_num = renew(buf_for_format, tmp_format);
 if(tmp_format_num == MALLOC_ERROR){
```

```
 return(MALLOC_ERROR);
}else if(tmp_format_num == ERROR){
 return(ERROR);
}

/* メインループ -- コメント行を飛ばして設定データ (1行) だけ読み出し */
for(;;){
 if(fgets(buf, sizeof(buf), fp) == NULL){
  printf("fgets error\n");
  return(FGETS_ERROR);
 }
 LOG((stderr, "fgets_buf = %s\n", buf));
 if(buf[0] == '#'){
 }else{
  /* 入力データを加工 ---> 可変引数のポインタがさす番地にデータ書き込み */
  /* buf 分解 */
  tmp_buf_num = renew(buf, tmp_buf);
  if(tmp_buf_num == MALLOC_ERROR){
   return(MALLOC_ERROR);
  }else if(tmp_buf_num == ERROR){
   return(ERROR);
  }
  LOG((stderr, "tmp_buf_num = %d(buf は%d 個に分解されました)\n", tmp_buf_num, (tmp_buf_num+1)));
  LOG((stderr, "tmp_format_num = %d(format は%d 個に分解されました)\n", tmp_format_num, (tmp_format_num+1)));
  LOG((stderr, "arg_num = %d(可変引数は%d 個です)\n", arg_num, arg_num));
  if(tmp_buf_num == tmp_format_num && (tmp_format_num+1) == arg_num){
   for(k=0;k<arg_num;k++){
    LOG((stderr, "tmp_buf[%d] = %s\n", k, tmp_buf[k]));
    LOG((stderr, "tmp_format[%d] = %s\n", k, tmp_format[k]));
    sscanf(tmp_buf[k], tmp_format[k], argument1[k]);
    free(tmp_format[k]);
    free(tmp_buf[k]);
   }
   free(buf_for_format);
   free(tmp_format);
   free(tmp_buf);
   free(argument1);
   return(SUCCESS);
  }
  /* 入力ファイルと引数の不一致の際の処理 */
  for(k=0;k<(tmp_buf_num+1);k++){
   free(tmp_buf[k]);
  }
  for(k=0;k<(tmp_format_num+1);k++){
   free(tmp_format[k]);
  }
  free(buf_for_format);
  free(tmp_format);
  free(tmp_buf);
  free(argument1);
  printf("入力ファイルと引数が一致しません\n");
  return(FORMAT_NOT_FIT_ARGUMENT_ERROR);
 }
}
}
```

●●● convert_data.h ●●●

```
#ifndef _CONVERT_DATA_H_INCLUDED_
#define _CONVERT_DATA_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) の output_data に出力用データを格納
 *
 * 【引数】
 * data 構造体 (make_ppm.h で定義)
 *
```

```
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 */
int convert_data(data *);

#endif /* _CONVERT_DATA_H_INCLUDED_ */
```

●●● convert_data.cpp ●●●

```
#include"convert_data.h"
#include<stdlib.h>

int convert_data(data *data_set){

 int i,k,x=0;
 int flg=0;

 /* メモリの割り当て */
 data_set->output_data = (unsigned char*)malloc(sizeof(unsigned int)*(data_set->col*data_set->row*3+1));
 if(data_set->output_data == NULL){
  printf("malloc data_set->output_data error\n");
  return(ERROR);
 }

 /* 出力データ作成 */
 for(i=0;i<(data_set->row*data_set->col);i++){
  for(k=0;k<data_set->div_num;k++){
   if(data_set->div_data[k] <= data_set->input_data[i] && data_set->input_data[i] < data_set->div_data[k+1] && flg != 1){
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[k][0];
    x++;
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[k][1];
    x++;
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[k][2];
    x++;
   }
   if(data_set->div_data[data_set->div_num] == data_set->input_data[i] && flg != 1){
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[data_set->div_num][0];
    x++;
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[data_set->div_num][1];
    x++;
    data_set->output_data[x] = (unsigned char)data_set->rgb_array[data_set->div_num][2];
    x++;
    flg++;
   }
  }
  flg = 0;
 }
 data_set->output_data[data_set->col*data_set->row*3] = '0';
 return(1);
}
```

●●● find_data.h ●●●

```
#ifndef _FIND_DATA_H_INCLUDED_
#define _FIND_DATA_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) の input_data(double*) から最大値を探し,data 構造体メンバの max_data に代入
 *
 * 【引数】
 * data 構造体 (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
```

```
 */
int find_max(data *);

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) の input_data(double*) から最小値を探し,data 構造体メンバの min_data に代入
 *
 * 【引数】
 * data 構造体 (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 */
int find_min(data *);

#endif /* _FIND_DATA_H_INCLUDED_ */
```

●●● find_data.cpp ●●●

```
#include"find_data.h"

int find_max(data *data_set){

 double tmp_max = 0.0;
 int i, j;

 for(i=0;i<(data_set->row*data_set->col);i++){
  if(data_set->input_data[i] > tmp_max){
   tmp_max = data_set->input_data[i];
  }
 }

 data_set->max_data = tmp_max;
// printf("data_set->max_data = %lf\n", data_set->max_data);

 return(1);
}

int find_min(data *data_set){

 double tmp_min = 0.0;
 int i, j;

 for(i=0;i<(data_set->row*data_set->col);i++){
  if(data_set->input_data[i] < tmp_min){
   tmp_min = data_set->input_data[i];
  }
 }

 data_set->min_data = tmp_min;
// printf("data_set->min_data = %lf\n", data_set->min_data);

 return(1);
}
```

●●● get_data.h ●●●

```
#ifndef _GET_DATA_H_INCLUDED_
#define _GET_DATA_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) のメンバが指し示すファイル (CSV 形式) からデータ (double 型) を取得
 * data_set->input_data[] に代入
 *
 * 【特徴】
 * 一行の長さに制限なし
```

```
 * #で始まるものはコメント行とみなす
 * 途中で#が来てもコメント行として扱う
 * ファイルの終端に改行が多数あっても問題は無い
 *
 * 【注意点】
 * コンマの数でデータ数を判別しているため，ファイルの終わりにもコンマは必要
 * 途中に空行があるファイルは読み込めない
 *
 * 【引数】
 * FILE *(すでにオープンしてあることが条件)，data 構造体ポインタ (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 */
int get_data(FILE *, data *);

#endif /* _GET_DATA_H_INCLUDED_ */
```

●●● get_data.cpp ●●●

```
#include"get_data.h"
#include<stdlib.h>

int get_data(FILE *fp_r, data *data_set){

 int i;
 int comma_num = 0, row_num = 1;
 int check_comment, comment_flg = 0, comma_num_tmp;
 char buf[3];

 /* ファイルポインタの位置指示子を作成 */
 fpos_t pos;

 /* realloc でループをまわすために一度メモリ確保 */
 data_set->input_data = (double *)malloc(sizeof(double)*1);
 if(data_set->input_data == NULL){
  printf("malloc data_set->input_data error\n");
  return(ERROR);
 }

 for(;;){

  /* コンマ数 (=データ数) のリセット */
  comma_num = 0;

  /* コメントフラグのリセット */
  comment_flg = 0;

  /* 現在のファイル位置指示子を保存 */
  fgetpos(fp_r, &pos);

  /* 行の長さをチェック */
  while((check_comment=fgetc(fp_r)) != '\n'){
   /* コメント行か否かをチェック (ascii 文字コード使用) */
   if(check_comment == 35){
    comment_flg = 1;
    break;
   }
   /* CSV 形式なので，コンマの数が数値の数 */
   if(check_comment == 44){
    comma_num++;
   }

   /* ファイルの終端：EOF */
   if(check_comment == EOF){
    break;
   }
  }
```

```
  /* EOF までくると次の行は comma_num == 0 && 空行もここで処理される */
  if(comma_num == 0 && comment_flg == 0){
   return(1);
  }

  if(comment_flg == 1){
   while((check_comment=fgetc(fp_r)) != '\n'){
    /* コメント行読み飛ばし */
   }
  }else{

   /* 各行の列数が異なる時の処理 */
   if(comma_num_tmp != comma_num && row_num != 1){
    printf("各行の列数が異なります\n");
    return(ERROR);
   }
   comma_num_tmp = comma_num;

   /* 現在のファイル位置指示子を戻す */
   fsetpos(fp_r, &pos);

   /* 行の成分数 (=コンマの数)*列数分の配列に realloc */
   data_set->input_data = (double *)realloc(data_set->input_data, sizeof(double)*(row_num*comma_num));
   if(data_set->input_data == NULL){
    printf("malloc data_set->input_data error\n");
    return(ERROR);
   }


   /* CSV データの読み込み */
   for(i=0;i<comma_num;i++){
    fscanf(fp_r, "%lf,", &(data_set->input_data[((row_num-1)*comma_num)+i]));
//    printf("data_set->input_data[%d] = %lf\n", i, data_set->input_data[((row_num-1)*comma_num)+i]);
   }

   /* 改行まで読み込み--->ファイル位置指示子を次の行へ */
   fgets(buf, sizeof(buf), fp_r);

   data_set->row = row_num;
   data_set->col = comma_num;
   row_num++;
  }
 }
}
```

●●● get_name.h ●●●

```
#ifndef _GET_NAME_H_INCLUDED_
#define _GET_NAME_H_INCLUDED_

/*
 * 【処理内容】
 * 入力されたファイル名から拡張子を取り除く
 * 代わりの拡張子として [第二引数の拡張子] を持ったファイル名 (入力ファイル名と同じ) を戻り値として返す
 *
 * 【引数】
 * char *(inputfile_name), char *(変更後の拡張子名)
 *
 * 【戻り値】
 * 正常：char *([malloc] してあります)
 * 異常：NULL
 *
 * 【備考】
 * 第二引数の拡張子は dot の有無によろず動作は保障される
 * ただし第二引数を間違って [..txt] などした場合の動作は保障しない
 */
char *get_name(char *, char *);

#endif /* _GET_NAME_H_INCLUDED_ */
```

●●● get_name.cpp ●●●

```
#include"get_name.h"
#include<stdio.h>
#include<stdlib.h>
#include <string.h>

char *get_name(char *input_name, char *extention){

 char *output_name;
 int dot_num=0, last_dot_flg, extention_len;
 int i=0;

 /* input_name から拡張子の部分まで [例 : test.txt なら test. まで] の文字数を取得 */
 while(input_name[i] != '\0'){
  if(input_name[i] == 46){/* ANCII 文字コード [.] */
   dot_num++;
   /* last_dot_flg は input_name 配列の何番目が最後の [.](拡張子のもの) かを示している */
   last_dot_flg = i;
  }
  i++;
 }

 extention_len = strlen(extention);

 /* 引数 [extention] が拡張子有りで渡されたときの処理 */
 if(extention[0] == 46){/* ANCII 文字コード [.] */
  for(i=0;i<extention_len;i++){
   extention[i] = extention[i+1];
  }
  extention_len -= 1;
 }

 /* 確認用 *//*
 printf("extention = %s\n", extention);
 printf("extention_len = %d\n", extention_len);
 */

 /* (last_dot_flg+1) ファイル名分+[ppm](3)+'\0'(1) 分 malloc */
 output_name = (char*)malloc(sizeof(char)*(last_dot_flg+1+extention_len+1));
 if(output_name == NULL){
  printf("malloc output_name error\n");
  return(NULL);
 }

 /* last_dot_flg は配列の何番目かを表しているので
  * プラス 1 個分入力ファイル名から出力ファイル名へコピーする
  */
 *strncpy(output_name, input_name, (size_t)(last_dot_flg+1));
 for(i=0;i<extention_len;i++){
  output_name[last_dot_flg+1+i] = extention[i];
 }
 output_name[last_dot_flg+1+extention_len] = '\0';

 /* 結果確認用 *//*
 printf("dot_num = %d\n", dot_num);
 printf("input_name = %s\n", input_name);
 printf("last_dot_flg = %d\n", last_dot_flg);
 printf("output_name = %s\n", output_name);
 */

 return(output_name);
}
```

●●● make_ppm.h ●●●

```
#ifndef _MAKE_PPM_H_INCLUDED_
#define _MAKE_PPM_H_INCLUDED_
```

```
#include<stdio.h>
#include"..\main\debug.h"

struct data{
 double max_data, min_data; /* 最大値・最小値 */
 double *input_data;        /* ファイルから読み込んだデータ配列 */
 double *div_data;          /* どこで分割するかを格納したデータ配列 */
 int **rgb_array;   /* 表示用RGBの配色データ格納配列 */
 unsigned char *output_data; /* 出力用配列 */
 int div_num;               /* 分割数 */
 int row;                   /* 列数 */
 int col;                   /* 行数 */
};

/*
 * 【処理内容】
 * 引数として渡された(char*)から入力ファイル名を取得しデータを読み込む ---> PPM(P6)画像形式で出力する
 * 出力ファイル名：引数のファイル名.ppm
 *
 * 【特徴】
 * 一行の長さに制限なし
 * #で始まるものはコメント行とみなす
 * 途中で#が来てもコメント行として扱う
 * ファイルの終端に改行が多数あっても問題は無い
 *
 * 【注意点】
 * コンマの数でデータ数を判別しているため，ファイルの終端(各行の終端)にもコンマは必要
 * 途中に空行があるファイルは読み込めない
 *
 * 【引数】
 * char *, int(表示用RGB配列の分割数)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 */
int make_ppm_p6(char *, int );

#endif /* _MAKE_PPM_H_INCLUDED_ */
```

●●● make_ppm.cpp ●●●

```
#include"get_data.h"
#include"find_data.h"
#include"convert_data.h"
#include"set_rgb.h"
#include"make_ppm.h"
#include"output.h"
#include"set_division.h"
#include"get_name.h"
#include<stdlib.h>

int make_ppm_p6(char *name, int div){

 int flg;
 data data_set;
 FILE *fp_r, *fp_w;
 char *output_file_name;

 /* 入力ファイルオープン */

 fp_r = fopen(name, "r");
 if(fp_r == NULL){
  printf("file open error\n");
  return(ERROR);
 }

 /* 出力ファイル名作成 */
 output_file_name = get_name(name, ".ppm");
```

```
 if(output_file_name == NULL){
  printf("error on get_output_file_name\n");
  return(ERROR);
 }

 /* 出力ファイルオープン */
 fp_w = fopen(output_file_name, "wb");
 if(fp_w == NULL){
  printf("file open error\n");
  return(ERROR);
 }

 /* ファイルからデータの読み込み */
 flg = get_data(fp_r, &data_set);
 if(flg == ERROR){
  printf("get_data error\n");
  return(ERROR);
 }

 /* ファイルからデータから最大値を取得 */
 find_max(&data_set);

 /* ファイルからデータから最小値を取得 */
 find_min(&data_set);

 /* 表示用 RGB 分割色の設定 */
 data_set.div_num = div;
 if(data_set.div_num < 10 || data_set.div_num > 100){
  printf("分割数が適正 (10 <= x <= 100) ではありません\n");
  printf("入力分割数:%d\n", data_set.div_num);
  return(ERROR);
 }
 set_rgb(&data_set);

 /* 分割用データ作成 */
 flg = set_division(&data_set);
 if(flg == ERROR){
  printf("get_data error\n");
  return(ERROR);
 }

 /* 出力用データ作成 */
 flg = convert_data(&data_set);
 if(flg == ERROR){
  printf("get_data error\n");
  return(ERROR);
 }

 /* データの書き出し */
 output(fp_w, &data_set);

 /* ファイルクローズ */
 fclose(fp_r);
 fclose(fp_w);

 /* free array */
 {
  int i;
  free(data_set.input_data);
  free(data_set.div_data);
  free(data_set.output_data);
  for(i=0;i<=data_set.div_num;i++){
   free(data_set.rgb_array[i]);
  }
  free(data_set.rgb_array);
 }

 return(1);
}
```

●●● output.h ●●●

```
#ifndef _OUTPUT_H_INCLUDED_
#define _OUTPUT_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data_set.output_data(unsigned char*型) をファイルに書き出す
 * 書き出すフォーマットは PPM(P6) 形式
 *
 * 【引数】
 * FILE *(出力用ファイルポインタ), data 構造体 (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 */
int output(FILE *, data *);

#endif /* _OUTPUT_H_INCLUDED_ */
```

●●● output.cpp ●●●

```
#include"output.h"

int output(FILE *fp_w, data *data_set){

 /* For #P6 */

 fprintf(fp_w, "P6\n");
 fprintf(fp_w, "#最大値:%lf 最小値:%lf 分割数:%d\n", data_set->max_data, data_set->min_data, data_set->div_num);
 fprintf(fp_w, "%d %d\n", data_set->col, data_set->row); /* col = width(横のピクセル数), row = lenght(縦のピク
セル数) */
 fprintf(fp_w, "255\n");
 fwrite(data_set->output_data, sizeof(unsigned char), (data_set->col*data_set->row*3), fp_w);

 return(1);
}
```

●●● set_division.h ●●●

```
#ifndef _SET_DIVISION_H_INCLUDED_
#define _SET_DIVISION_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) の div_data に値 (下記のような) を代入
 *
 * ・値の例
 *   最小値:0
 *   最大値:10
 *   分割数:10
 *   配列に入る値
 *   div_data[0] = 0
 *   div_data[1] = 1
 *   div_data[2] = 2
 *   div_data[3] = 3
 *   div_data[4] = 4
 *   div_data[5] = 5
 *   div_data[6] = 6
 *   div_data[7] = 7
 *   div_data[8] = 8
 *   div_data[9] = 9
 *   div_data[10] = 10
 *
```

```
 * +----+----+----+----+----+----+----+----+----+----+
 * 0    1    2    3    4    5    6    7    8    9   10
 +
 * 【引数】
 * data 構造体 (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 * 【備考】
 * 10 分割ならば 11 個のデータ (植木算)
 */
int set_division(data *);

#endif /* _SET_DIVISION_H_INCLUDED_ */
```

●●● set_division.cpp ●●●

```
#include"set_division.h"
#include<stdlib.h>

int set_division(data *data_set){

 double sa, div_data_tmp;
 int i;

 sa = data_set->max_data - data_set->min_data;

 data_set->div_data = (double*)malloc(sizeof(double)*(data_set->div_num+1));
 if(data_set->div_data == NULL){
  printf("malloc data_set->div_data error\n");
  return(ERROR);
 }

 div_data_tmp = sa/data_set->div_num;

 data_set->div_data[0] = data_set->min_data;
 data_set->div_data[data_set->div_num] = data_set->max_data;

 for(i=1;i<data_set->div_num;i++){
  data_set->div_data[i] = data_set->min_data + div_data_tmp * (double)i;
//  printf("data_set->div_data[%d] = %lf\n", i, data_set->div_data[i]);
 }

 return(1);
}
```

●●● set_rgb.h ●●●

```
#ifndef _SET_RGB_H_INCLUDED_
#define _SET_RGB_H_INCLUDED_

#include"make_ppm.h"

/*
 * 【処理内容】
 * data 構造体 (make_ppm.h で定義) の rgb_array に表示用の RGB データを格納
 * 色の決定は,data->div_num によって変化 (div_num >= 2 が条件) だが,10 以上を推奨
 *
 * 【言葉の定義】
 * 1 分割
 * +----+
 * 0    1
 *
 * 2 分割
 * +----+----+
 * 0    1    2
 *
 * 【引数】
```

```
 * data 構造体 (make_ppm.h で定義)
 *
 * 【戻り値】
 * 正常：1
 * 異常：ERROR
 *
 * 【備考】
 * 10 分割ならば 11 個のデータ (RGB) が出来る (植木算)
 */
int set_rgb(data *);

#endif /* _SET_RGB_H_INCLUDED_ */
```

●●● set_rgb.cpp ●●●

```
#include"set_rgb.h"
#include<stdlib.h>

int set_rgb(data *data_set){

 int i;
 double x;

 data_set->rgb_array = (int**)malloc(sizeof(int*)*(data_set->div_num+1));
 if(data_set->rgb_array == NULL){
  printf("malloc data_set->rgb_array error\n");
  return(ERROR);
 }
 for(i=0;i<=(data_set->div_num+1);i++){
  data_set->rgb_array[i] = (int*)malloc(sizeof(int)*(3));
  if(data_set->rgb_array[i] == NULL){
   printf("malloc data_set->rgb_array[i] error\n", i);
   return(ERROR);
  }
 }

 /* data_set->div_num >= 10 が条件 */
 if(data_set->div_num < 10){
  printf("data_set->div_num error\n");
  return(ERROR);
 }

 /* 1 番目の RGB 値 */
 data_set->rgb_array[0][0]=0;
 data_set->rgb_array[0][1]=0;
 data_set->rgb_array[0][2]=255;

 /* (div_num+1) 番目の RGB 値 */
 data_set->rgb_array[data_set->div_num][0]=255;
 data_set->rgb_array[data_set->div_num][1]=0;
 data_set->rgb_array[data_set->div_num][2]=0;

 x = 100.0/(double)(data_set->div_num);

 for(i=1;i<data_set->div_num;i++){

  double y;

  y = i*x;

  if(0.0 <= y && y < 49.0){
   data_set->rgb_array[i][0]=(int)0.0;
   data_set->rgb_array[i][1]=(int)(5.1*y);
   data_set->rgb_array[i][2]=(int)(-5.1*(y)+255.0);
  }
  else if(49.0 <= y && y < 100.0){
   data_set->rgb_array[i][0]=(int)(5.1*(y-49.0));
   data_set->rgb_array[i][1]=(int)(-5.1*(y-49.0)+255.0);
   data_set->rgb_array[i][2]=(int)0.0;
  }
```

```
 }

 /* 代入結果の表示 */
 /*
 {
 FILE *test;
 test = fopen("test_rgb.txt", "w");
 if(test == NULL){
  printf("file open error\n");
  return(ERROR);
 }
 for(i=0;i<=data_set->div_num;i++){
  printf("%3d %3d %3d\n", data_set->rgb_array[i][0], data_set->rgb_array[i][1], data_set->rgb_array[i][2]);
  fprintf(test, "%3d %3d %3d\n", data_set->rgb_array[i][0], data_set->rgb_array[i][1], data_set->rgb_array[i][2]);
 }
 fclose(test);
 }
 */
 return(1);
}
```

参考文献

[1] 宇野亨, ”FDTD 法による電磁界およびアンテナ解析 ” コロナ社 (1998)

[2] K.S.Yee, "Numerical Solution of Value Problems Involving Maxwell's Equationsin Isotropic Media", IEEE Trans.Antennas Propag.,14,4,pp.302-307,May,1966.

第5章　結言

本研究では, 電磁界シミュレーション技術を高周波デバイス (高周波トランス) の設計に応用することによって, 従来にない広帯域な伝送機器の開発を念頭におき,3 つの研究を行なった. それは次のとおりである.

まずは CATV に用いられる通信デバイスである分配器をターゲットとし, その動作周波数帯域の広帯域化, ひいては分配器の構成要素である高周波トランスの広帯域化を目指した. この研究により広帯域での高周波トランスの設計指針を得ることが出来た. 特に広帯域に渡って, 交差巻きが特性改善に大きな影響を与えるという事実を明らかにし, その知見をもとに試作したデバイスについて 10〜2600MHz において損失 3dB 以下という優れた特性を得ることが出来た.

次に,CATV 用の二次歪み補償方式として提案されている P-P 伝送方式システムにおける P-P トランスの動作特性について解析を行なった. またその結果をもとに歪み補償システム全体の性能を評価した. シミュレーション結果は測定結果とよい一致を見たが, 詳細な比較検討は今後の課題として残った.

最後に C 言語による FDTD プログラムの開発を行なった. この目的は高周波トランスの巻き線を交差巻きにしても, 特性が悪化するものなどがある原因を時間応答から検討するためである. 原因究明が出来ることで, 高周波トランス設計の新たな設計指針を得ることが出来, 従来から高周波トランスの設計で行なわれている職人による勘, 経験などと言った属人的な要素を減らし, 製作者によらず同様の優れた特性を有するデバイス開発出来ると考えられる. ただし現段階では, 基礎的なソルバーが完成したにとどまる. プログラムの実用は今後の課題である.

謝辞

本研究にあたり, 多大なるご指導, ご助言をいただきました竹尾隆教授, 野呂雄一准教授に心からの感謝の意を表します. また, 本研究に対し貴重なご助力をいただいた山本好弘技官, 本講座の院生ならびに学部生諸氏に感謝いたします.