平成17年度
卒業研究報告書

有限積分法による
電磁場の時間領域解析

秋田工業高等専門学校　電気工学科

研究者名　　滑川　雅人

指導教員名　　山本　昌志

要旨

本研究では有限積分法を用いて，時間領域の電磁場を解析するためのプログラム作成を行った．解析対象は軸対称周期構造の加速管である．

有限積分法では 2 種類の Yee 格子と呼ばれる空間格子で空間を分割する．その空間におけるマクスウェル方程式はマクスウェルグリッド方程式と呼ばれる式で表される．

この有限積分法を基に，軸対称加速管内を伝搬する TM_0 モードの電磁場を計算する基本式の導出を行いプログラムを作成した．

目 次

第1章　序論

加速器などの設計では，計算機による数値シミュレーションは実用上非常に有用である．加速器に限らず，どのような機器の設計でも，実物を製作して所定の特性を有しているかを試験することは，時間とコストの面からみて著しく効率の悪い方法である．また，簡単なモデルについては人間が特性を算出できるが，複雑なモデルについては事実上不可能である．そのため，計算機による数値シミュレーションは非常によく用いられている．

現在，マイクロ波やアンテナ伝搬などの3次元時間領域におけるコンピュータを用いた数値シミュレーションでは，差分型の方法である時間領域差分法(FDTD法)や有限積分法(FIT法)が主に用いられている[1]．この2つは，ともに1966年にYeeによって考案された，Yee格子と呼ばれる空間格子モデルで，電磁場を空間と時間に関して離散化するという概念に基づいている．FDTD法は時間領域の解析にのみ用いることができるが，FIT法は時間領域，周波数領域，静電磁場，渦電流場に適用が可能である．本研究では，広範囲に応用が可能なFIT法を用いて，軸対称加速管内部の電磁場を数値シミュレーションする計算プログラムの開発を行なった．

今回は以下に示す軸対称周期構造の加速管を解析対象とした．

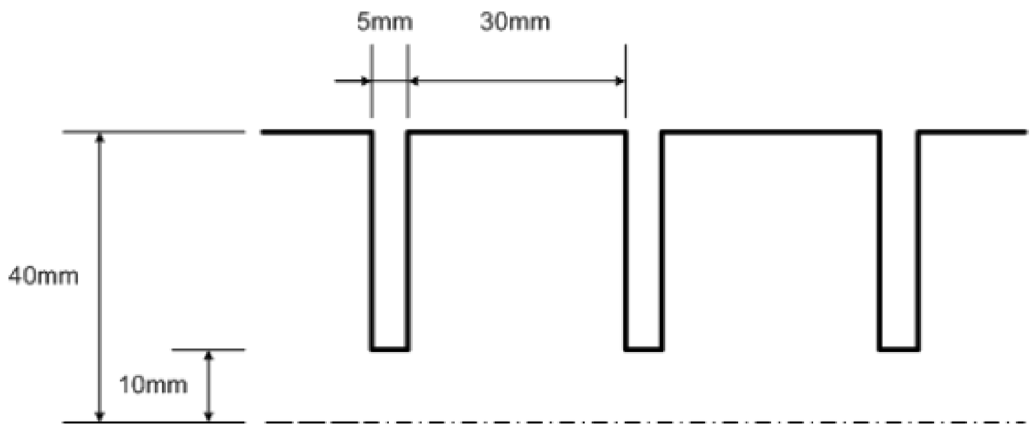

図 1.1: 軸対称周期構造加速管

この加速管の共振周波数は約3000MHzである．

第2章　有限積分法による電磁場解析理論

2.1　FITの基本概念

電流も電荷も存在しない空間 ($\rho = 0$, $\boldsymbol{j} = 0$) における電磁場は，マクスウェル方程式

$$\nabla \cdot \boldsymbol{D} = 0 \tag{2.1}$$

$$\nabla \cdot \boldsymbol{B} = 0 \tag{2.2}$$

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{2.3}$$

$$\nabla \times \boldsymbol{H} = \frac{\partial \boldsymbol{D}}{\partial t} \tag{2.4}$$

で表される．ここで，$\boldsymbol{D}$ は電束密度，$\boldsymbol{B}$ は磁束，$\boldsymbol{E}$ は電場，$\boldsymbol{H}$ は磁場である．また，

$$\boldsymbol{D} = \varepsilon \boldsymbol{E} \tag{2.5}$$

$$\boldsymbol{B} = \mu \boldsymbol{H} \tag{2.6}$$

ここで，ε は空間の誘電率，μ は空間の透磁率である．これらの偏微分方程式をコンピュータで直接解くことはできないが，マクスウェル方程式を差分化すれば解くことができる．そのためにはまず，時間と空間を離散化する必要がある．ここで，時間と空間をそれぞれ $\triangle t, \triangle l$ を単位に離散化する（ここでは，簡単のため x, y, z すべての方向で，単位格子サイズ $\triangle l$ で離散化するものとした）[2].

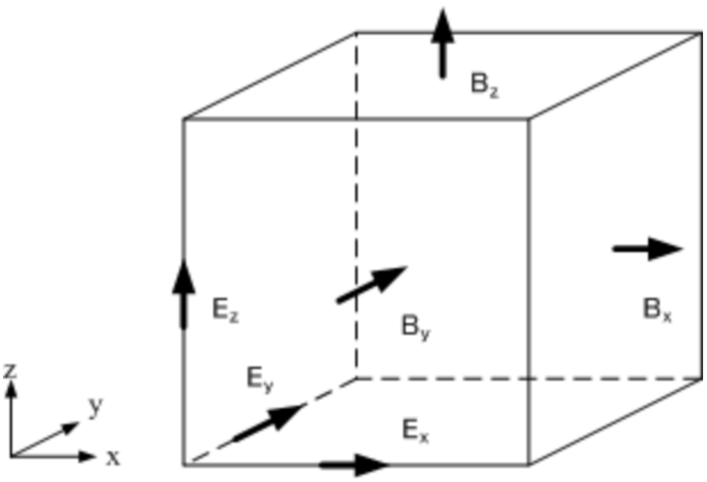

図 2.1: 電磁場の Yee 格子モデル

図 2.1 の，空間格子に電磁場の各成分を配置したものは Yee 格子と呼ばれる．この Yee 格子を 2 種類に分け，図 2.2 の 2 重グリッドで空間を格子化するというのが FIT 法の基本的な概念である．図 2.2 において，グリッド $\boldsymbol{G}$ 上に $\boldsymbol{E}, \boldsymbol{B}$，デュアルグリッド $\tilde{\boldsymbol{G}}$ 上に $\boldsymbol{D}, \boldsymbol{H}$ を配置している．ここで，デュアルグリッド $\tilde{\boldsymbol{G}}$ の頂点がグリッド G の中心にくるように配置されている．

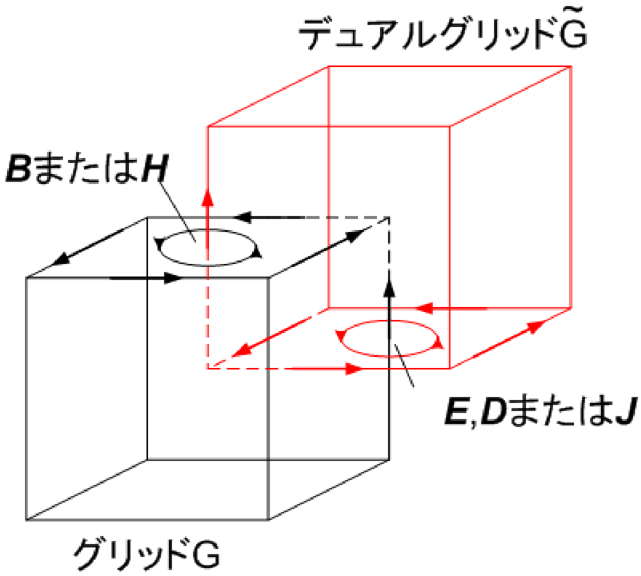

図 2.2: 電磁場の 2 重グリッド構造

2.2　マクスウェルグリッド方程式

ここでは，マクスウェル方程式の回転の項のみを取り扱うこととする．式 (2.3) と式 (2.4) の両辺を dS で積分すると

$$\int \nabla \times \boldsymbol{E} \cdot \boldsymbol{n}\, dS = -\frac{\partial}{\partial t} \int \boldsymbol{B} \cdot \boldsymbol{n}\, dS \tag{2.7}$$

$$\int \nabla \times \boldsymbol{H} \cdot \boldsymbol{n}\, dS = \frac{\partial}{\partial t} \int \boldsymbol{D} \cdot \boldsymbol{n}\, dS \tag{2.8}$$

となる．式 (2.7) と式 (2.8) にストークスの定理を適用して，

$$\int \nabla \times \boldsymbol{E} \cdot \boldsymbol{n}\, dS = \int \boldsymbol{E}\, d\ell \tag{2.9}$$

$$\int \nabla \times \boldsymbol{H} \cdot \boldsymbol{n}\, dS = \int \boldsymbol{H}\, d\ell \tag{2.10}$$

と変形できる．よって，式 (2.7) と式 (2.8) は

$$\oint \boldsymbol{E}\ d\ell = -\frac{\partial}{\partial\, t}\int \boldsymbol{B}\cdot\boldsymbol{n}\ dS \tag{2.11}$$

$$\oint \boldsymbol{H}\ d\ell = \frac{\partial}{\partial\, t}\int \boldsymbol{D}\cdot\boldsymbol{n}\ dS \tag{2.12}$$

と表される．ここで，図 2.3 のようなグリッド面を考える．

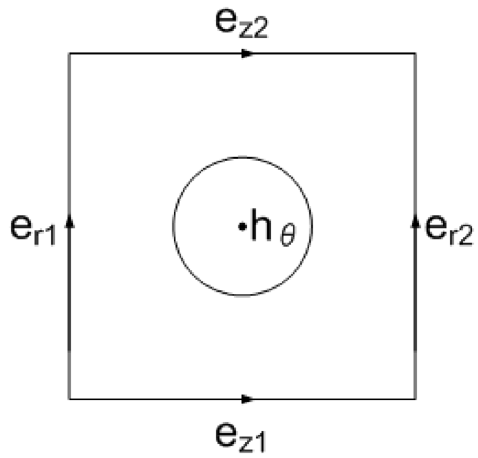

図 2.3: ファラデーの周回積分の有限積分表示

このとき，

$$\boldsymbol{e} = \int \boldsymbol{E}\ d\ell \tag{2.13}$$

$$\boldsymbol{b} = \int \boldsymbol{B}\cdot\boldsymbol{n}\ dS \tag{2.14}$$

$$\boldsymbol{h} = \int \boldsymbol{H}\ d\ell \tag{2.15}$$

$$\boldsymbol{d} = \int \boldsymbol{D}\cdot\boldsymbol{n}\ dS \tag{2.16}$$

と定義すると，図 2.3 上での式 (2.11) と式 (2.12) は

$$e_{y2} - e_{y1} - e_{x2} + e_{x1} = \dot{b}_z \tag{2.17}$$

$$h_{y2} - h_{y1} - h_{x2} + h_{x1} = \dot{d}_z \tag{2.18}$$

で表される．このように，方程式の係数が $1, -1, 0$ で表される．これを領域全体について考えると，各グリッドの方程式の連立方程式になる．ここで，電場を各辺で積分した量 $\boldsymbol{e}$，磁束を各面で積分した量 $\boldsymbol{b}$，磁場を各辺で積分した量 $\boldsymbol{h}$，電束密度を各面で積分した量 $\boldsymbol{d}$

を電磁場の各成分を 1 列に並べた行列とすると

$$\boldsymbol{e} = [e_{x1}, e_{x2}, \cdots e_{xn}|e_{y1}, e_{y2}, \cdots e_{yn}|e_{z1}, e_{z2}, \cdots e_{zn}]^t \tag{2.19}$$

$$\boldsymbol{b} = [b_{x1}, b_{x2}, \cdots b_{xn}|b_{y1}, b_{y2}, \cdots b_{yn}|b_{z1}, b_{z2}, \cdots b_{zn}]^t \tag{2.20}$$

$$\boldsymbol{h} = [h_{x1}, h_{x2}, \cdots h_{xn}|h_{y1}, h_{y2}, \cdots h_{yn}|h_{z1}, h_{z2}, \cdots h_{zn}]^t \tag{2.21}$$

$$\boldsymbol{d} = [d_{x1}, d_{x2}, \cdots d_{xn}|d_{y1}, d_{y2}, \cdots d_{yn}|d_{z1}, d_{z2}, \cdots d_{zn}]^t \tag{2.22}$$

となる．式 (2.19)，式 (2.20)，式 (2.21)，式 (2.22) を用いた各グリッドにおける式 (2.17)，式 (2.18) の連立方程式は

$$C\ \boldsymbol{e} = -\dot{\boldsymbol{b}} \tag{2.23}$$

$$\tilde{C}\ \boldsymbol{h} = \dot{\boldsymbol{d}} \tag{2.24}$$

と書ける．式 (2.23)，式 (2.24) はマクスウェルグリッド 方程式と呼ばれる．ここで，$C, \tilde{C}$ は成分が 1，-1，0 で構成される行列である．式 (2.3)，式 (2.4) と式 (2.23)，式 (2.24) を比較すると，$C, \tilde{C}$ は rot 演算子に対応している．今回，我々は直に使用しなかったが，発散 (div 演算子) に対応する行列 S と $\tilde{S}$ もある．すなわち，有限積分法ではマクスウェル方程式に表れる回転や発散は，それと 1 対 1 に対応する行列に置き換わる．このことから，自己矛盾のない完全な離散化されたマクスウェル方程式ができる．

2.3　解析対象

今回，我々が解析対象としたのは図 1.1 に示す周期構造の軸対称加速管である．境界は金属境界と両端が吸収境界，内部空洞は真空とし，電流も電荷も存在しないものとした．さらに，取り扱うモードは TM_0 とする．すると，位置 (r, z, θ) の関数である電場の各成分 (E_r, E_z, E_θ) と磁場の各成分 (H_r, H_z, H_θ) のうち

$$E_\theta = 0 \qquad H_r = 0 \qquad H_z = 0 \tag{2.25}$$

となる．また，その対称性から E_r, E_z, H_θ は θ に依存せず，(r, z) の関数となる．

2.4　軸対称問題

以上のことから，軸対称問題におけるマクスウェルグリッド 方程式について考える．ここで，位置 (r, z) と時間 (t) の関数である電磁場の各成分を簡潔に表記するため，以下のように記述する．

$$e(r, z, t) = e^t_{r,z} \tag{2.26}$$

$$b(r, z, t) = b^t_{r,z} \tag{2.27}$$

$$h(r, z, t) = h^t_{r,z} \tag{2.28}$$

$$d(r, z, t) = d^t_{r,z} \tag{2.29}$$

2.4.1　rz 平面

式 (2.23) を図 2.4 の領域について考える．まず，この領域において式 (2.11) の右辺と左辺はそれぞれ

$$
\begin{aligned}
\oint \boldsymbol{E}\, d\ell &= \int E_{z_{i,j-\frac{1}{2}}}^{n+\frac{1}{2}}\, d\ell - \int E_{z_{i,j+\frac{1}{2}}}^{n+\frac{1}{2}}\, d\ell \\
&\quad - \int E_{r_{i-\frac{1}{2},j}}^{n+\frac{1}{2}}\, d\ell + \int E_{z_{i+\frac{1}{2},j}}^{n+\frac{1}{2}}\, d\ell \\
&= e_{z_{i,j-\frac{1}{2}}}^{n+\frac{1}{2}} - e_{z_{i,j+\frac{1}{2}}}^{n+\frac{1}{2}} \\
&\quad - e_{r_{i-\frac{1}{2},j}}^{n+\frac{1}{2}} + e_{z_{i+\frac{1}{2},j}}^{n+\frac{1}{2}} \qquad (2.30)
\end{aligned}
$$

$$
-\frac{\partial}{\partial\, t}\int \boldsymbol{B}\cdot\boldsymbol{n}\, dS = -\dot{b}_{\theta i,j} \qquad (2.31)
$$

となる．式 (2.30) と式 (2.31) より，式 (2.23) の行列 C は $b_{\theta i,j}$ と同じ行の $e_{zi,j-\frac{1}{2}}, e_{zi,j+\frac{1}{2}}, e_{ri-\frac{1}{2},j}, e_{ri+\frac{1}{2},j}$ に対応する列に 1 または -1 を配置し，それ以外の列には 0 を配置すればよい．

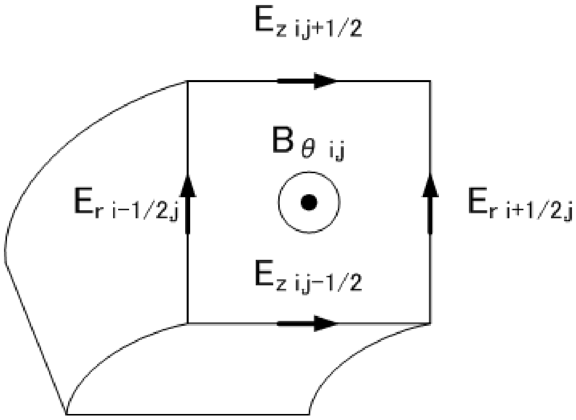

図 2.4: rz 平面

2.4.2　θz 平面

次に，式 (2.24) を図 2.5 の領域について考える．まず，この領域において式 (2.12) の右辺と左辺はそれぞれ

$$
\begin{aligned}
\oint \boldsymbol{H}\, d\ell &= \int H_{\theta_{i-1,j}}^{n}\, d\ell - \int H_{\theta_{i,j}}^{n}\, d\ell \\
&= h_{\theta_{i-1,j}}^{n} - h_{\theta_{i,j}}^{n} \qquad (2.32)
\end{aligned}
$$

$$
\frac{\partial}{\partial\, t}\int \boldsymbol{D}\cdot\boldsymbol{n}\, dS = \dot{d}_{r i-\frac{1}{2},j} \qquad (2.33)
$$

となる．式 (2.32) と式 (2.33) より，rz 平面の場合と同様に，式 (2.24) の行列 $\tilde{C}$ は $d_{r_{i-\frac{1}{2}},j}$ と同じ行の $e_{\theta i-1,j}, e_{\theta i,j}$ に対応する列に 1 または -1 を配置し，それ以外の列には 0 を配置すればよい．

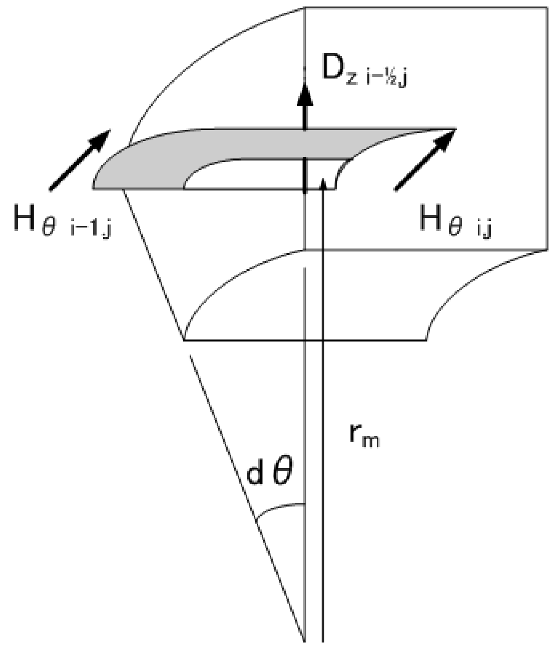

図 2.5: θz 平面

2.4.3 $r\theta$ 平面

同様に，式 (2.24) を図 2.6 の領域についても考える．まず，この領域において式 (2.12) の右辺と左辺はそれぞれ

$$\begin{aligned}\oint \boldsymbol{H}\ d\ell &= \int H_{\theta_{i,j+1}}^{n}\ d\ell - \int H_{\theta_{i,j}}^{n}\ d\ell \\ &= h_{\theta_{i,j+1}}^{n} - h_{\theta_{i,j}}^{n} \end{aligned} \tag{2.34}$$

$$\frac{\partial}{\partial t}\int \boldsymbol{D}\cdot\boldsymbol{n}\ dS = \dot{d}_{z_{i,j+\frac{1}{2}}} \tag{2.35}$$

となる．式 (2.34) と式 (2.35) より，rz 平面，θz 平面の場合と同様に，式 (2.24) の行列 $\tilde{C}$ は $d_{z_{i,j+\frac{1}{2}}}$ と同じ行の $e_{\theta i,j+1}, e_{\theta i,j}$ に対応する列に 1 または -1 を配置し，それ以外の列には 0 を配置すればよい．

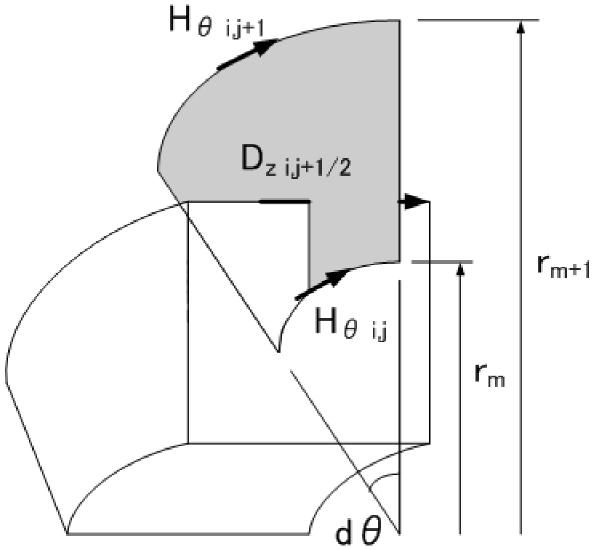

図 2.6: $r\theta$ 平面

2.5　軸対称問題の有限積分法における電磁場の相互関係

前節で，軸対称問題における行列 $C, \tilde{C}$ の成分の配置について述べた．しかし，実際に式 (2.23) と式 (2.24) を解くためには，$\boldsymbol{e}$ と $\boldsymbol{d}$，$\boldsymbol{b}$ と $\boldsymbol{h}$ それぞれの相互関係を知る必要がある．ここでは，$\boldsymbol{e}$ と $\boldsymbol{d}$，$\boldsymbol{b}$ と $\boldsymbol{h}$ それぞれの相互関係について述べる．

2.5.1　$\boldsymbol{e}$ と $\boldsymbol{d}$ の相互関係

電場 $\boldsymbol{E}$ と電束密度 $\boldsymbol{D}$ の間には式 (2.5) の関係がある．ここで，領域中で ε は一定とする．まず，r 成分について考える．図 2.5 において

$$e_r = \int E_r \, d\ell = E_r \, d\ell \tag{2.36}$$

$$d_r = \int D_r \, dS = \varepsilon \int E_r \, dS = \varepsilon E_r \, d\ell \; r_m d\theta \tag{2.37}$$

である．式 (2.36) と式 (2.37) より

$$d_r = \varepsilon e_r \; r_m d\theta \tag{2.38}$$

となる．ここで，r_m は中心からの距離である．

次に，z 成分について考える．図 2.6 において

$$e_z = \int E_z \, d\ell = E_z \, d\ell \tag{2.39}$$

$$d_z = \int D_z \, dS = \varepsilon \int E_z \, dS = \varepsilon E_z \, d\ell \; \frac{1}{2} \left({r_{m+1}}^2 - {r_m}^2 \right) d\theta \tag{2.40}$$

となる．式 (2.39) と式 (2.40) から

$$d_z = \varepsilon e_z \frac{1}{2}\left({r_{m+1}}^2 - {r_m}^2\right) d\theta \tag{2.41}$$

と書ける．ここで，r_{m+1}, r_m は中心からの距離である．

2.5.2　*b* と *h* の相互関係

$\boldsymbol{b}$ と $\boldsymbol{h}$ についても同様に考える．磁束 $\boldsymbol{B}$ と磁場 $\boldsymbol{H}$ には式 (2.6) の関係がある．ここで，領域内で μ は一定とする．図 2.4 において

$$b_\theta = \int B_\theta \ dS = B_\theta \ d\ell^2 \tag{2.42}$$

$$h_\theta = \int H_\theta \ d\ell = \frac{1}{\mu}\int B_\theta \ d\ell = \frac{1}{\mu} B_\theta \ r_m d\theta \tag{2.43}$$

である．式 (2.42) と式 (2.43) より

$$h_\theta = \frac{1}{\mu \ d\ell^2} b_\theta \ r_m d\theta \tag{2.44}$$

となる．

2.6　軸対称問題におけるマクスウェルグリッド 方程式

式 (2.38)，式 (2.41)，式 (2.44) より式 (2.24) を書き直すと

$$\left(\frac{1}{\mu \ d\ell^2} \ r_m d\theta\right) \tilde{C} \ b_\theta = \left(r_m \ \text{又は} \ \frac{1}{2}{r_{m+1}}^2 - {r_m}^2\right)(\varepsilon \ d\theta)\,\dot{\boldsymbol{e}} \tag{2.45}$$

式 (2.45) を変形して，

$$(A_r \ \text{又は} \ A_z)\,\tilde{C} \ b_\theta = \dot{\boldsymbol{e}} \tag{2.46}$$

$$A_r = \left(\frac{1}{\varepsilon\mu \ d\ell^2}\right) \tag{2.47}$$

$$A_z = \left(\frac{1}{\varepsilon\mu \ d\ell^2}\right)\left(\frac{2r_m}{{r_{m+1}}^2 - {r_m}^2}\right) \tag{2.48}$$

となる．式 (2.46) において各係数を行列 $\tilde{C}$ 内に含むことにすると

$$\hat{C} \ \boldsymbol{b} = \dot{\boldsymbol{e}} \tag{2.49}$$

と書ける．ここで，$\hat{C}$ は $\tilde{C}$ の e_r に対応する列に A_r を，e_z に対応する列に A_z を乗算したものである．

2.7　数値計算に用いる方程式

式 (2.23) と式 (2.49) の右辺は微分の定義から

$$-\dot{\boldsymbol{b}} = -\frac{\boldsymbol{b}^n - \boldsymbol{b}^{n-1}}{dt} \tag{2.50}$$

$$\dot{\boldsymbol{e}} = \frac{\boldsymbol{e}^{n+\frac{1}{2}} - \boldsymbol{e}^{n-\frac{1}{2}}}{dt} \tag{2.51}$$

である．ここで，dt は微少な時間の刻み幅である．式 (2.50) と式 (2.51) から，式 (2.23) と式 (2.49) は

$$C\ \boldsymbol{e}^{n-\frac{1}{2}} = -\frac{\boldsymbol{b}^n - \boldsymbol{b}^{n-1}}{dt} \tag{2.52}$$

$$\hat{C}\ \boldsymbol{b}^n = \frac{\boldsymbol{e}^{n+\frac{1}{2}} - \boldsymbol{e}^{n-\frac{1}{2}}}{dt} \tag{2.53}$$

となる．式 (2.52) と式 (2.53) を変形すると

$$\boldsymbol{b}^n = \boldsymbol{b}^{n-1} - C\ \boldsymbol{e}^{n-\frac{1}{2}}\ dt \tag{2.54}$$

$$\boldsymbol{e}^{n+\frac{1}{2}} = \boldsymbol{e}^{n-\frac{1}{2}} + \hat{C}\ \boldsymbol{b}^n\ dt \tag{2.55}$$

と書ける．式 (2.54) と式 (2.55) を交互に計算することで，時間を追って電磁場の状態を算出することができる．ここで，安定した解を得るための条件として

$$dt \leq \frac{1}{\sqrt{\left(\frac{1}{dx}\right)^2 + \left(\frac{1}{dy}\right)^2 + \left(\frac{1}{dz}\right)^2}}\frac{1}{C_0} \tag{2.56}$$

という条件（クーラン条件）が知られている．dx, dy, dx は xyz 座標系における，Yee 格子の x 方向，y 方向，z 方向のサイズである．C_0 は光の速度（$C_0 = 299792458$ [m/s]）である．

第3章　数値計算

3.1　プログラムの概要

本研究では，有限積分法を用いて電磁場を時間領域で解析するプログラムを作成した．使用した言語はC++である．

プログラムの構成は主に次の構成から成っている．

1. クーラン条件から時間刻み幅を決定
2. 行列 $C, \hat{C}$ の成分配置
3. 格子の座標を記述したファイルの出力
4. 式 (2.54) から磁束を算出する
5. 式 (2.55) から電場を算出する
6. 境界条件を適用 [3]
7. 計算結果の出力

また，今回作成したプログラムとは別に出力したファイルから，電磁場の状態をグラフィカルに表示するプログラムを用いた．

まず，計算領域と格子サイズを入力する．そして，クーラン条件から時間の刻み幅を決定する．また，計算領域と境界条件から式 (2.47) と式 (2.48) を用いて行列 $C, \hat{C}$ の成分を計算する．そして，境界条件を適用しながら式 (2.54) と式 (2.55) を交互に計算することで，電磁場の時間領域における状態の変化を解析できる．

第4章　計算結果

今回作成したプログラムを用いて，図1.1の軸対称加速管にパルスを入射したときの電磁場の状態を解析した．以下に解析結果を示す．図は磁場の強さを表示している．図において緑は0, 黄色から赤，白と変化するにつれ正に大きな磁場，青から黒へと変化するにつれ負に大きな磁場を表す．ここで，パルス入射時間 $T = 3[nS]$，時間刻み幅 $dt = 2.358595 \times 10^{-12}[S]$ ですべて一定である．

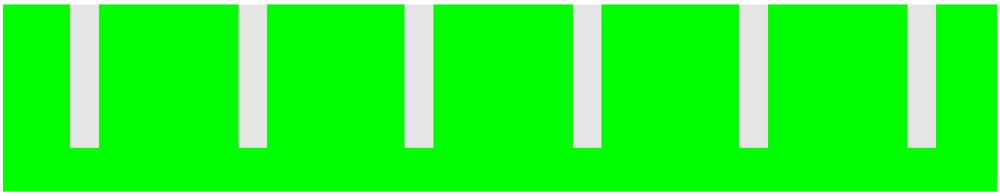

図 4.1: 初期状態

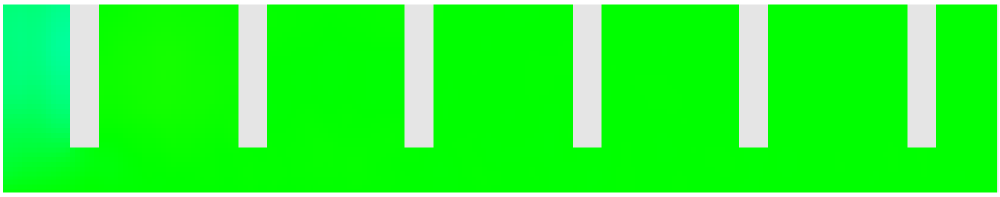

図 4.2: 入射信号 0.6GHz　計算ステップ 450Step

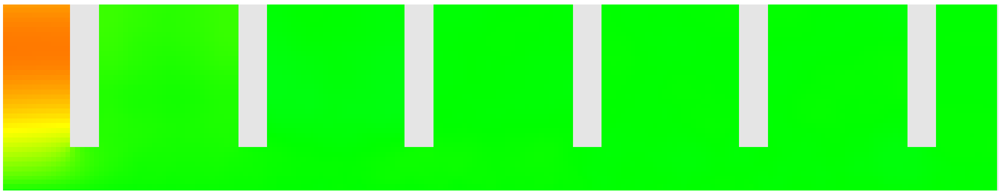

図 4.3: 入射信号 0.6GHz　計算ステップ 1500Step

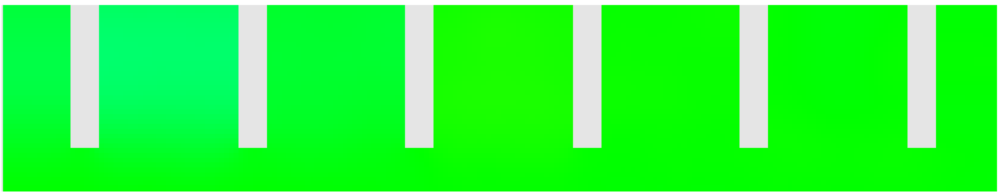

図 4.4: 入射信号 0.6GHz　計算ステップ 4500Step

共振周波数からある程度低い周波数になると，入り口付近でほとんど反射し，右側へは電磁場は通過しない．

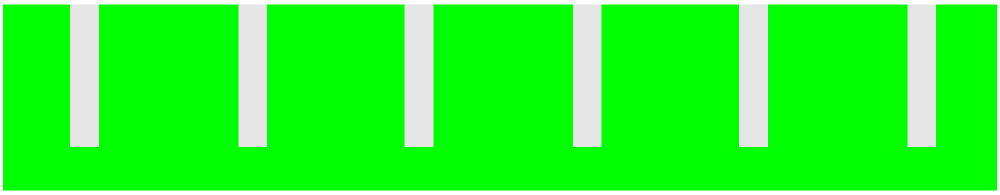

図 4.5: 初期状態

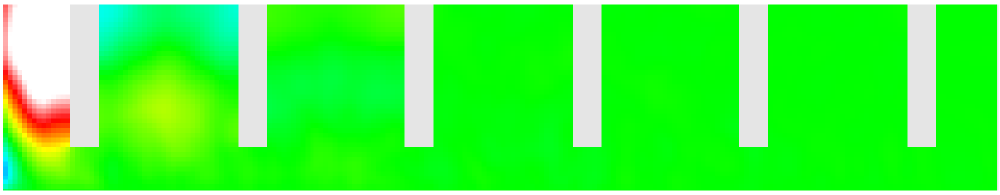

図 4.6: 入射信号 6GHz　計算ステップ 450Step

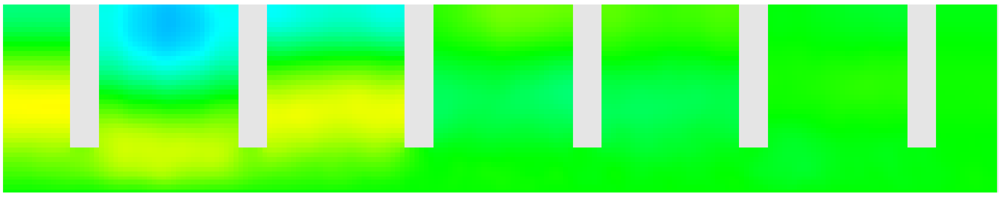

図 4.7: 入射信号 6GHz　計算ステップ 1500Step

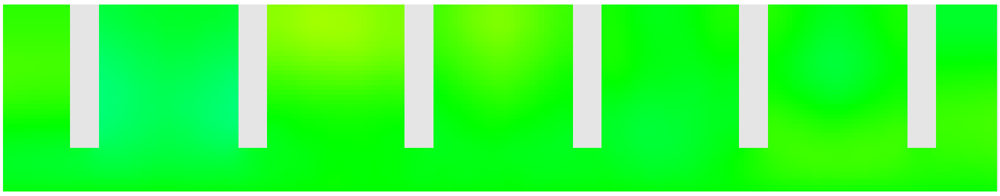

図 4.8: 入射信号 6GHz　計算ステップ 4500Step

共振周波数よりもある程度大きい周波数では，電磁場は加速管の内部にほとんど残らずに抜けていくことがわかる．

このことから，この加速管は周波数フィルタの機能を持つことがわかる．

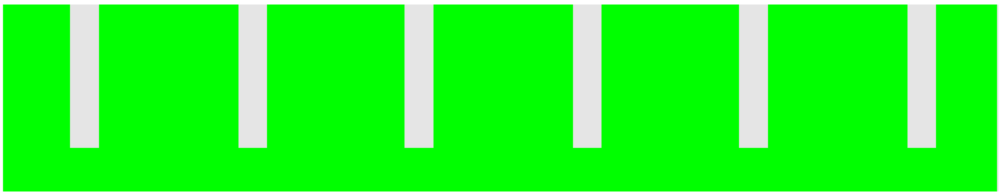

図 4.9: 初期状態

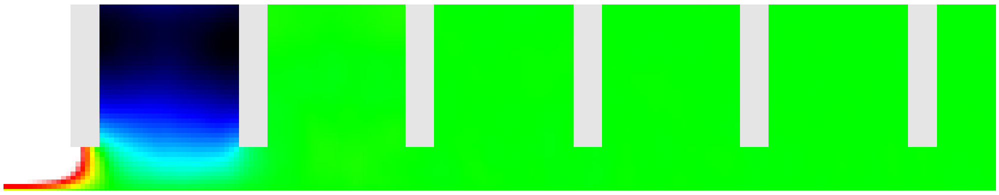

図 4.10: 入射信号 3GHz　計算ステップ 450Step

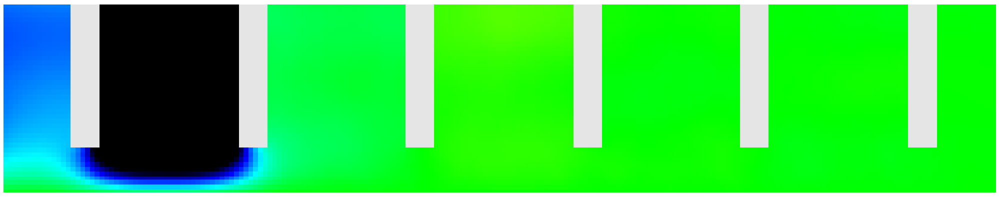

図 4.11: 入射信号 3GHz　計算ステップ 1500Step

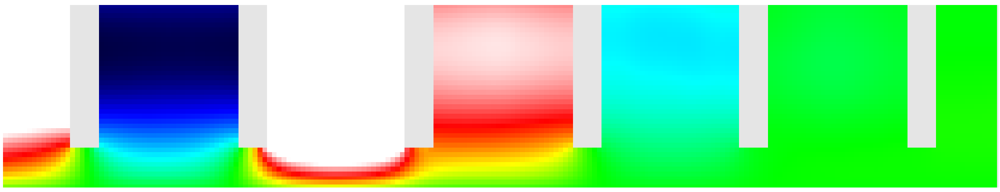

図 4.12: 入射信号 3GHz　計算ステップ 4500Step

共振周波数付近では，電磁場の進行速度が低下し，エネルギー密度が増加している．そのため，強い電磁場が加速管内に蓄積していることがわかる．

加速管はこの蓄積された強い電磁場で粒子を加速する機器である．

第5章　まとめ

軸対称空間に対する有限積分法の計算式を導いた．その計算式を用いて，有限積分法による軸対称加速管内部の電磁場を時間領域解析するプログラムを作成した．

今後の課題として，まずは計算精度の検証と精度の向上が挙げられる．今回の研究では計算精度に関しては検証しなかった．しかし，計算精度は加速器の設計などでは非常に重要である．そのため，計算精度を検証する事とその向上が望まれる．具体的な検証方法については，市販されているプログラムとの比較が考えられる．

また，三角形メッシュへの対応が挙げられる．今回作成したプログラムでは領域を正方形分割したが，これは形状精度があまりよくない．すなわち，曲線要素や複雑な形状に対してよい分割を行うことができない．そのため，より形状精度の高い三角形メッシュへの対応も望まれる．これは計算精度の向上にもつながる．

さらに，計算速度の向上も考えられる．本研究室では，作成したプログラムを最終的には商品化する事を目指している．その際，計算速度が速いことは一つのセールスポイントと成りうる．加えて，計算速度が向上すればより細かいメッシュや，より広い領域での解析を行うことが可能となる．そのため，計算速度を向上させることも必要である．

関連図書

[1] 本間利久・五十嵐一・川口秀樹. 計算電気・電子工学シリーズ 数値電磁力学. 森北出版株式会社, 2002.

[2] R. Wanzenberg T. Weiland. Wake field and impedances, may 1991. DESY M-91-06.

[3] 堀之内總一他. ANSI C による　数値計算法入門. 森北出版株式会社, 2002.

謝辞

ご指導してくださった担当教員の山本昌志先生，さまざまな面でご協力していただいた専攻科の夏井拓也さん，宮田翔吾さんに感謝の意を表します．

付 録A　計算プログラム

リスト A.1: 有限積分法を用いた軸対称加速管内電磁場解析プログラム

```
#include<iostream>
using namespace std;
#include<cmath>
#include<fstream>
#include<cstdlib>
#include<cstring>
#include "matrix.h"

void FIT();
int Courant(double &dlt, double DELTA_L, double C);

void Cal_const(double &dlr, double &dlz, double &ct1, double &ctb,
double &cte, double C, double DELTA_L,
double MU, double UPSILON, double dlt);

void Make_C_Matrix(int nez, int ner, int nr, int nb, Matrix &ce,
Matrix &cb, int Height[], int Width[]);
void File_init(char filename[], int nz, int nr, int MAX_FILE,
double Z, double R, int Height[], int Width[]);
void ABC(int nez, int ne, int nr, double ct1, Matrix &e, Matrix e_ret);
void BC(int nr, int nez, Matrix &e, double t, double freq);
void Make_wall(int nez, int nr, int nz, int ner, int nez,
Matrix &e, int Height[], int Width[]);
int File_out(char filename[], int nr, int nz, int nez, int ne,
Matrix e, Matrix b);

void Cal_mag(Matrix e, Matrix &b, Matrix ce, double cte, int nb);
void Ret_ele(Matrix e, Matrix &e_ret, int ne, int nr, int nez);
void Cal_ele(Matrix &e, Matrix b, Matrix cb, int ne, int nez,
int nr, double R0, double dlr, double ctb);

int main(void){

  FIT();
  return 0;

}

void FIT(){
  const double R = .210;
  const double R0 = .000;
  const double Z = .210;
  const double DELTA_L = .001;
  const double DELTA_T = 1.0e-10;
```

```
    const double Height_in = .001;
    const double Height_high = .040 + Height_in;
    const double Height_low = .010 + Height_in;
    const double Width_init = .015;
    const double Width_wide = .030;
    const double Width_narrow = .005;
    const double T1 = 3.0e-9;
    //const double T1 = 10.0e-6;

    const double UPSILON = 8.85418e-12;
    const double MU = 1.25663e-6;
    const double C = 2.998e8;

    const int MAX_FILE = 999;
    const int OUT_STEP = 15;

    double r2 = 0.0;
    double r1 = 0.0;
    double rr = 0.0;
    double dlt = 0.0;

    int nr = int ( ( R - R0 ) / DELTA_L );
    int nz = int ( Z / DELTA_L );
    int nglid = nr * nz;
    int nez = nglid + nz;
    int ner = nglid + nr;
    int ne = nez + ner;
    int nb = nglid + nr;

    int flag = 0;
    int i = 0;
    int j = 0;
    int k = 0;
    int temp_i = 0;
    double t = 0.0;

    int Height[3];
    int Width[3];

    Height[0] = int ( Height_in / DELTA_L );
    Height[1] = int ( Height_low / DELTA_L );
    Height[2] = int ( Height_high / DELTA_L );

    Width[0] = int ( Width_init / DELTA_L );
    Width[1] = int ( Width_narrow / DELTA_L );
    Width[2] = int ( Width_wide / DELTA_L );

    double dlr = 0.0;
    double dlz = 0.0;
    double ct1 = 0.0;
    double ctb = 0.0;
    double cte = 0.0;
    double freq = 0.0;
    char filename[80] = "file_out";
    char temp_c;
```

```

  double sum = 0.0;
  int sign = 0;
  int num = 0;

  //Make Matrix
  Matrix e(ne, 1);
  printf("Make Matrix e is ok!!\n");

  Matrix b(nb, 1);
  printf("Make Matrix b is ok!!\n");

  Matrix ce(nb, 5);
  printf("Make Matrix ce is ok!!\n");

  Matrix cb(ne, 5);
  printf("Make Matrix cb is ok!!\n");

  Matrix e_ret(ne, 1);
  printf("Make Matrix ok!!\n");

  //Inspection of Courant condition
  Courant(dlt, DELTA_L, C);

  //Calculation of constant
  Cal_const(dlr, dlz, ct1, ctb, cte, C, DELTA_L, MU, UPSILON, dlt);

  //Input of freqency
  printf("How much is freqency ?\n");
  printf("f MHz = ");
  scanf("%lf%c", &freq, &temp_c);
  printf("f MHz = %lf\n", freq);

  //File_init
  File_init(filename, nz, nr, MAX_FILE, Z, R, Height, Width);

  //Make C matrix
  Make_C_Matrix(nez, ner, nr, nb, ce, cb, Height, Width);

  //Decision of territory
  //Dec_ter(nr, nz, nez, ne, nb, ce, cb, Height, Width);

  while(1){
    //i = 1;
    //j = 1;

    //Calculation of magnetic field
    Cal_mag(e, b, ce, cte, nb);

    //Retention of electric field
    //er
    Ret_ele(e, e_ret, ne, nr, nez);

    //Calculation of electric field
    Cal_ele(e, b, cb, ne, nez, nr, R0, dlr, ctb);
```

```
    
    //Absorption boundary condition
    ABC(nez, ne, nr, ct1, e, e_ret);

    //Boundary condition
    if(t <= T1){
      BC(nr, nez, e, t, freq);
    }
    Make_wall(nez, nr, nz, ner, nez, e, Height, Width);

    //file_out
    if( 0 == flag ){
      if( File_out(filename, nr, nz, nez, ne, e, b) >= MAX_FILE ){
      break;
      }
      //cout << t << "\n";
    }

    t += dlt;
    flag++;
    if( OUT_STEP <= flag ){
      flag = 0;
    }
  }
}

int Courant(double &dlt, double DELTA_L, double C){
 dlt = DELTA_L / sqrt(2) / C;
 return 1;
}

void Cal_const(double &dlr, double &dlz, double &ct1, double &ctb,
double &cte, double C, double DELTA_L, double MU, double UPSILON,
double dlt){

  dlr = DELTA_L;
  dlz = DELTA_L;
  ct1 = (C * dlt - DELTA_L) / (C * dlt + DELTA_L);
  //cte = dlt;
  //ctb = dlt / ( DELTA_L * DELTA_L * MU * UPSILON);
  ctb = dlt / UPSILON;
  //cte = dlt / MU / DELTA_L;
  cte = dlt / ( DELTA_L * DELTA_L * MU );
  //ctb = dlt / UPSILON;
  //cte = dlt / MU;



  printf("dlt = %e\tct1 = %e\n", dlt, ct1);
  printf("cte = %e\tctb = %e\n", cte, ctb);
}

void Make_C_Matrix(int nez, int ner, int nr, int nb, Matrix &ce,
Matrix &cb, int Height[], int Width[]){
  int i = 0;
```

```
    int count = 1;

    //ez
    for(i = 1; i <= nez; i++){
      if( Height[2]+3 < ( i % nr ) ) continue;
      count = 1;
      switch(i % nr + 1){
      case 0:
        break;
        /*
      case 1:
        ce[i][count] = i;
        count++;
        ce[i][count] = -(-i);
        count++;
        break;
        */
      default:
        ce[i][count] = i;
        count++;
        ce[i][count] = -(i + 1);
        count++;
        break;
      }
    //er
      if(i <= ner-nr){
        ce[i][count] = -(nez + i);
        count++;
        ce[i][count] = nez + i + nr;
      }
    }

    //hs
    count = 1;
    //ez
    for(i = 1; i <= nb - nr; i++){
      if( Height[2]+3 < ( i % nr ) ) continue;
      switch(i % nr){
      case 1:
        cb[i][1] = -(-i);
        cb[i][2] = i;
        break;
      default:
        //if(1 != i % nr){
        cb[i][1] = -(i - 1);
        cb[i][2] = i;
        //}
        break;
      }
    }
    //er
    for(i = nr+1; i <= nb - nr; i++){
      if( Height[2] < ( i % nr ) ) continue;
      cb[nez+i][1] = i - nr;
      cb[nez+i][2] = -i;
```

```
  }
  cout << "Make C Matris OK!!\n";

}


void File_init(char filename[], int nz, int nr, int MAX_FILE,
double Z, double R, int Height[], int Width[]){
  int file_out_r = 0;
  int file_out_z = 0;
  double file_out_rl = 0.0;
  double file_out_zl = 0.0;
  double dr = R / nr;
  double dz = Z / nz;
  int flag = 2;
  int count = 0;
  int check = 2;

  FILE *fp;
  fp = fopen(filename, "w");

  fprintf(fp, "$options\n");
  fprintf(fp, "horizontal_range\t0\t%lf\n", Z);
  fprintf(fp, "vertical_range\t0\t%lf\n", R);
  fprintf(fp, "value_range\t-0.1\t0.1\n");
  fprintf(fp, "max_data %d\n\n", MAX_FILE);
  fprintf(fp, "$end\n");
  fprintf(fp, "$coordinates\n");

  for(file_out_z = 1; file_out_z <= (nz); file_out_z++){
    file_out_zl = file_out_z * dz;

    for(file_out_r = 1; file_out_r <= (nr); file_out_r++){
      file_out_rl = file_out_r * dr;
      //fprintf(fp, "%lf\t%lf\tpaint\n", file_out_zl, file_out_rl);

      if(Height[0] <= file_out_r && Height[check] > file_out_r){
        fprintf(fp, "%lf\t%lf\tpaint\n", file_out_zl, file_out_rl);
      }
      else{
        fprintf(fp, "%lf\t%lf\tnot\n", file_out_zl, file_out_rl);
      }

    }

    count++;

    if(2 == flag && Width[0] <= count){
      flag = !flag;
      check = 1;
      count = 0;
    }
    else if(0 == flag && Width[1] < count){
      flag = !flag;
      check = 2;
```

```
      count = 0;
    }
    else if(1 == flag && Width[2]-1 <= count){
      flag = !flag;
      check = 1;
      count = 0;
    }

    fprintf(fp, "\n");
  }
  fclose(fp);
  printf("file_init ok!!\n");
}

void ABC(int nez, int ne, int nr, double ct1, Matrix &e, Matrix e_ret){
  int i;
    //er
    for(i = 1; i <= (nr); i++){
      e[ne-nr+i][1] = e_ret[ne-2*nr+i][1]
+ ct1 * (e[ne-2*nr+i][1] - e_ret[ne-nr+i][1]);

      e[nez+i][1] = e_ret[nez+nr+i][1]
+ ct1 * (e[nez+nr+i][1] - e_ret[nez+i][1]);

      //e[nez+i+nr][1] = e_ret[nez+nr+i+nr][1]
+ ct1 * (e[nez+nr+i+nr][1] - e_ret[nez+i+nr][1]);

      //e[nez+i+nr*4][1] = e_ret[nez+nr+i+nr*4][1]
+ ct1 * (e[nez+nr+i+nr*4][1] - e_ret[nez+i+nr*4][1]);
    }
}

void BC(int nr, int nez, Matrix &e, double t, double freq){
  int i;
  double f;
  f = freq * t * 1000000.0 * M_PI * 2.0;
  for(i = 1; i <= nr; i++){
  //for(i = nr + 1; i <= (nr * 2); i++){
  //for(i = (nr*5 + 1); i <= (nr * 6); i++){
    e[nez+i][1] = .05 * sin(f);
  }
}

void Make_wall(int enz, int nr, int nz, int ner, int nez,
Matrix &e, int Height[], int Width[]){
  int h_in = Height[0];
  int h_low = Height[1];
  int h_high = Height[2];
  int w_init = Width[0];
  int w_narrow = Width[1];
  int w_wide = Width[2];
  int h = 0;
  int w = 0;

  h = h_high;
```

```
  w = w_init;

  int i = 0;
  int j = 0;
  int k = 0;
  int count = 0;
  int flag = 0;

  /*
  for(i = 1; i < h_in; i++){
    e[nez+i+nr][1] = 0.0;
  }
  */
  for(i = h_high; i <= nr; i++){
    e[nez+i][1] = 0.0;
  }
  /*
  for(i = 0; i <= nz; i++){
    e[enz+1+i*nr][1] = 0.0;
  }
  */
  /*
  for(i = 0; i <= nz; i++){
    e[enz+h_in+i*nr][1] = 0.0;
  }
  */
  for(i = 0; i <= nz-1; i++){

    if(0 == flag && count >= w){

      for(h = h_high; h >= h_low; h--){
        e[nez+h+i*nr][1] = 0.0;
      }

      flag = !flag;
      h = h_low;
      w = w_wide;
      count = 0;
    }

    if(1 == flag && count >= w_narrow){

      for(h = h_low; h <= h_high; h++){
        e[nez+h+i*nr][1] = 0.0;
      }

      flag = !flag;
      h = h_high;
      count = 0;
    }

    e[h+i*nr][1] = 0.0;
    count++;
  }

```

```
}

int File_out(char filename[], int nr, int nz, int nez,int ne,
Matrix e, Matrix b){
  FILE *fp;

  static int i = 0;
  static int j = 0;
  static int k = 0;
  static int n = 0;

  int r = 0;
  int z = 0;
  char temp_c[80];

  strcpy (temp_c, filename);

  n++;
  i++;
  if(10 <= i){
    j++;
    if(10 <= j){
      k++;
      j = 0;
    }
    i = 0;
  }

  for(r = 0; r <= 2; r++){
    if(0 == r){
      z = k;
    }
    else if(1 == r){
      z = j;
    }
    else if(2 == r){
        z = i;
    }
    else{
      printf("error!\n");
    }

    switch(z){
    case 0:
      if(!(0 == k && 0 == r || 0 == k && 0 == j && 1 == r))
        {
          strcat(filename, "0" );
        }
      break;

    case 1:
      strcat(filename, "1" );
      break;

    case 2:
```

```
      strcat(filename, "2" );
      break;

    case 3:
      strcat(filename, "3" );
      break;

    case 4:
      strcat(filename, "4" );
      break;

    case 5:
      strcat(filename, "5" );
      break;

    case 6:
      strcat(filename, "6" );
      break;

    case 7:
      strcat(filename, "7" );
      break;

    case 8:
      strcat(filename, "8" );
      break;

    case 9:
      strcat(filename, "9" );
      break;

    default:
      break;
    }
  }

  int count = 0;
  fp = fopen(filename, "w");
  for(z = 1; z <= (nz); z++){
    for(r = 1; r <= (nr); r++){
      count++;
      //fprintf(fp, "%lf\n", e[nez+count][1]);
      fprintf(fp, "%lf\n", b[count][1]);
    }
    fprintf(fp, "\n");
  }
  fclose(fp);
  cout << filename << "\n";
  strcpy (filename, temp_c);
  return n;
}

void Cal_mag(Matrix e, Matrix &b, Matrix ce, double cte, int nb){
  int i = 1;
  int j = 1;
```

```
  int sign = 0;
  int num = 0;
  double sum = .0;

    for(i = 1; i <= nb; i++){

      while( ce[i][j] ){
        num = int ( fabs ( ce[i][j] ) );
        sign = int( ce[i][j] / num );
        sum += sign * e[num][1];
        j++;
      }

      b[i][1] -= cte * sum;
      j = 1;
      sum = 0.0;
    }

}

void Ret_ele(Matrix e, Matrix &e_ret , int ne, int nr, int nez){
  int i = 1;

    for(i = 1; i <= (nr); i++){
      e_ret[ne-2*nr+i][1] = e[ne-2*nr+i][1];
      e_ret[ne-nr+i][1] = e[ne-nr+i][1];

      e_ret[nez+2*nr+i][1] = e[nez+2*nr+i][1];
      e_ret[nez+nr+i][1] = e[nez+nr+i][1];
      e_ret[nez+i][1] = e[nez+i][1];
    }
}

void Cal_ele(Matrix &e, Matrix b, Matrix cb, int ne, int nez,
int nr, double R0, double dlr, double ctb){
  int i = 1;
  int j = 1;
  int k = 1;
  double r1 = .0;
  double r2 = .0;
  double rr = .0;
  int num = 0;
  int sign = 0;
  double sum = .0;

      for(i = 1; i <= ne; i++){

        //if(i > nez){
        if(i <= nez){
          /*
          if(1 == ((i - nez) % (nr + 1))){
            k = 20;
          }
          else{
            r1 = dlr * (i%nr) + R0;
```

```
          r2 = dlr * (i%nr+1) + R0;
          k = 10;
        }
        */

        r1 = dlr * (i%nr) + R0;
        r2 = dlr * (i%nr+1) + R0;
        k = 10;

      }
      else{
        k = 0;
      }

      while(cb[i][j]){
        num = int ( fabs ( cb[i][j] ) );
        sign = int ( cb[i][j] / num );

        switch(j+k){
        case 11:
          rr = 2.0 * r1 / (r2 + r1);
          //cout << rr << "\n";
          break;
        case 12:
          rr = 2.0 * r2 / (r2 + r1);
          //cout << rr << "\n";
          break;
        case 21:
        case 22:
          //rr = dlr / 2.0 * M_PI;
          rr = dlr * M_PI;
          //cout << rr << "\n";
          break;
        default:
          rr = 1.0;
          break;
      }

        //cout << rr << "\n";
        sum += sign * b[num][1] * rr;
        //sum += sign * b[num][1];
        j++;
      }
    e[i][1] += ctb * sum;
    j = 1;
    sum = 0.0;
  }
}
```