平成16年度
卒業研究報告書

有限要素法による
高周波共振モード解析の基礎的研究

秋田工業高等専門学校　電気工学科

研究者名　　佐々木　亨

指導教員名　　山本　昌志

要旨

本研究では、有限要素法を用いて高周波共振モードの解析を行うためのプログラムを作成した。解析対象は、加速空洞などの共振空洞である。このような空洞を設計する場合に、空洞内部の電磁場や共振周波数を計算することが大変重要となる。

空洞内の電磁場は、マクスウェルの方程式によって記述される。空洞内は真空であり、また電荷や電流は存在しないという条件の下では、それはヘルムホルツ方程式

$$\nabla^2 \boldsymbol{H} + \lambda^2 \boldsymbol{H} = 0$$

になる。 $\boldsymbol{H}$ は磁場を表し空間の関数、 λ は固有値である。解析する領域は軸対称構造であるから、上式の微分演算子に円柱座標系を適用する必要がある。さらに、解析するモードを単極の TM モードとすると、その汎関数は

$$F[H_\theta] = \int \left[\left(\frac{\partial H_\theta}{\partial z} \right)^2 + \left(\frac{\partial H_\theta}{\partial r} \right)^2 + 2 \frac{H_\theta}{r} \frac{\partial H_\theta}{\partial r} + \left(\frac{H_\theta}{r} \right)^2 - \lambda^2 H_\theta^2 \right] 2\pi r dr dz$$

となる。実際に有限要素法を用いて計算する場合は、計算領域を三角形のメッシュで区切る。そして、上に示した汎関数を各三角形要素に適用しながら計算する。このようにして離散化された汎関数の第 1 変分を 0 とする式は

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$$

という形の一般化固有値問題に還元される。ここではこの一般化固有値問題を共役勾配法によって計算し、H_θ および λ を求めた。

作成したプログラムを用いて KEK の加速空洞の共振モードを計算した。計算の結果、磁場の分布を求めることができた。また、計算により求められた共振周波数は 500.0MHz であった。測定結果は 499.5MHz であったので、よい精度で計算できていることがわかる。

本研究において、有限要素法により共振モードが計算できるようになったが、さらに精度を上げるため 2 次要素の導入を考える必要がある。また、定在波のみならず進行波の問題を取り扱えるようになると、その応用範囲が広がる。

目 次

第1章　序論

加速空洞のような共振空洞を設計する場合、その共振モードの解析は重要である。コンピュータシミュレーションによりその共振モードの周波数と電磁場分布を計算し、空洞形状の最適化を行うことになる。本研究では、軸対称構造の共振空洞内における定在波問題、すなわち固有振動数および空洞内に分布する電磁場の強度を有限要素法で求める基礎的な研究を行い、プログラムを作成した。

今回は、以下に示す加速空洞の解析を行った。

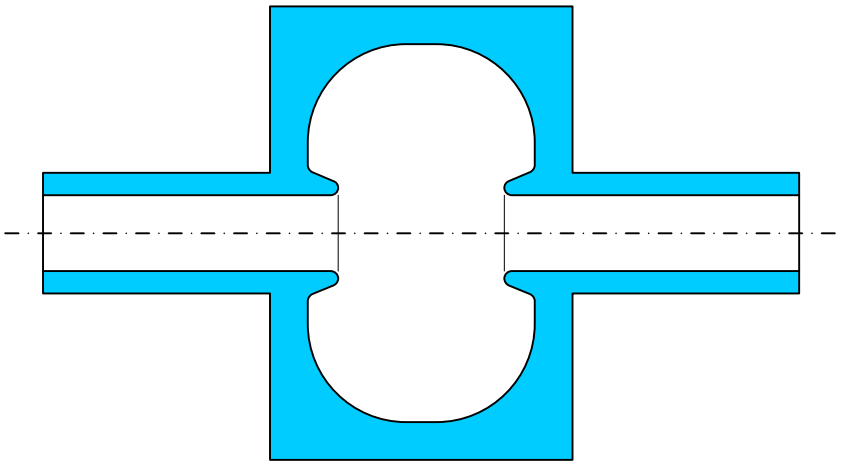

図 1.1: 軸対称共振空洞（KEK の PF の空洞）

この空洞は直径約 47cm で、軸対称構造になっている。また、共振周波数は 500MHz 程度である。

第2章　有限要素法による電磁場解析の理論

2.1　マクスウェルの方程式

ここでは軸対称構造の共振空洞内の共振モードの電磁場の方程式を示す。この場合、内部は真空で、金属で囲まれた空間になる。当然、ここの電磁場はマクスウェルの方程式で記述される。ただし、内部には電荷も電流が無いという条件が付され、$\rho = 0, \boldsymbol{j} = 0$ となる。

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

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

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

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

また、誘電率と透磁率は一定で、それぞれ ε_0, μ_0 となる。そして、

$$\boldsymbol{D} = \varepsilon_0 \boldsymbol{E} \tag{2.5}$$

$$\boldsymbol{B} = \mu_0 \boldsymbol{H} \tag{2.6}$$

の関係がある。式 (2.1)〜(2.4) は連立の偏微分方程式なので、計算しやすい形に直す。

まず、磁場の方程式を求める。そのために、式 (2.4) の両辺に回転の演算子を作用させる。そうすると、

$$\begin{aligned} \nabla \times \nabla \times \boldsymbol{H} &= \nabla \times \frac{\partial \boldsymbol{D}}{\partial t} \\ &= \frac{\partial}{\partial t} \left(\nabla \times \varepsilon_0 \boldsymbol{E} \right) \\ &= -\varepsilon_0 \frac{\partial}{\partial t} \left(\frac{\partial \boldsymbol{B}}{\partial t} \right) \\ &= -\mu_0 \varepsilon_0 \frac{\partial^2 \boldsymbol{H}}{\partial t^2} \end{aligned} \tag{2.7}$$

となる。これは、波動方程式である。電磁波の速度は光速 c で、この方程式では

$$\mu_0 \varepsilon_0 = \frac{1}{c^2} \tag{2.8}$$

となる。従って、磁場 H が満たす方程式は、

$$\nabla \times \nabla \times \boldsymbol{H} = -\frac{1}{c^2} \frac{\partial^2 \boldsymbol{H}}{\partial t^2} \tag{2.9}$$

となる。

この式も、場所と時間の両方の項の偏微分方程式なので、解くのは困難である。そのため、

$$\boldsymbol{H}(\boldsymbol{r},t)=\boldsymbol{H}(\boldsymbol{r})f(t) \tag{2.10}$$

と変数分離ができるか考える。式 (2.10) を (2.9) に入れると

$$\nabla\times\nabla\times\{\boldsymbol{H}(\boldsymbol{r})f(t)\}=-\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\{\boldsymbol{H}(\boldsymbol{r})f(t)\} \tag{2.11}$$

となり、時間と空間の微分を考えると

$$f(t)\nabla\times\nabla\times\{\boldsymbol{H}(\boldsymbol{r})\}=-\boldsymbol{H}(\boldsymbol{r})\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\{f(t)\} \tag{2.12}$$

となる。以降、簡素に記述するために、磁場の空間の関数を $\boldsymbol{H}(\boldsymbol{r})$ は $\boldsymbol{H_r}$ と、時間の関数 $f(t)$ は f_t とする。変数分離をするので、左辺と右辺を時間及び空間のみの関数にしたいわけだが、ベクトルの演算ということを考慮し、この式の両辺に $\boldsymbol{H_r}/(\boldsymbol{H_r}\cdot\boldsymbol{H_r})$ なるベクトルの内積の演算を施す。すると

$$f_t\frac{\boldsymbol{H_r}}{\boldsymbol{H_r}\cdot\boldsymbol{H_r}}\cdot\{\nabla\times\nabla\times\boldsymbol{H_r}\}=-\frac{\boldsymbol{H_r}\cdot\boldsymbol{H_r}}{\boldsymbol{H_r}\cdot\boldsymbol{H_r}}\frac{1}{c^2}\frac{\partial^2 f_t}{\partial t^2} \tag{2.13}$$

となる。これを整理すると、

$$\frac{\boldsymbol{H_r}}{\boldsymbol{H_r}\cdot\boldsymbol{H_r}}\cdot\{\nabla\times\nabla\times\boldsymbol{H_r}\}=-\frac{1}{f_t}\frac{1}{c^2}\frac{\partial^2 f_t}{\partial t^2} \tag{2.14}$$

である。この偏微分方程式は、左辺は空間 $\boldsymbol{r}$、右辺は時間 t のみの関数である。それぞれ別の独立変数となっているので、この等式が成り立つためには、その値は定数でなくてはならない。この定数を ω^2/c^2 とする[1]。そうすると、

$$\frac{\boldsymbol{H_r}}{\boldsymbol{H_r}\cdot\boldsymbol{H_r}}\cdot\{\nabla\times\nabla\times\boldsymbol{H_r}\}=-\frac{1}{f_t}\frac{1}{c^2}\frac{\partial^2 f_t}{\partial t^2}=\frac{\omega^2}{c^2} \tag{2.15}$$

である。2 番目と 3 番目の式から、時間のみの微分方程式がつくられ、それはもはや偏微分方程式ではなく、常微分方程式

$$\frac{d^2 f_t}{dt^2}=-\omega^2 f_t \tag{2.16}$$

になる。この微分方程式は、簡単に解けて

$$f_t=ae^{-i\omega t+\theta_0} \tag{2.17}$$

となる。ここで、a と θ_0 は初期条件により決まる定数である。これで、変数分離した解 (2.10) の時間の項が求まった。この時間の項は、三角関数になっている。

残りの空間の項 $\boldsymbol{H_r}$ について考えなくてならない。それが満たす偏微分方程式を得るために、時間の項の結果である式 (2.17) を、式 (2.12) に適用する。すると、

$$ae^{-i\omega t+\theta_0}\left[\nabla\times\nabla\times\boldsymbol{H_r}\right]=-\boldsymbol{H_r}\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\left(ae^{-i\omega t+\theta_0}\right) \tag{2.18}$$

[1] ω^2/c^2 と 2 乗にするのは、後の計算が簡単になるからである。ω を複素数の範囲まで考えると、その定数を a とするのと同じである。

となる。時間の項の微分を行うと、

$$ae^{-i\omega t+\theta_0}\left[\nabla\times\nabla\times\boldsymbol{H_r}\right]=\boldsymbol{H_r}\frac{\omega^2}{c^2}\left(ae^{-i\omega t+\theta_0}\right) \tag{2.19}$$

となる。即ち時間の微分 $(\partial/\partial t)$ は、$-i\omega$ に置き換えられるのである。さらに整理すると、

$$\nabla\times\nabla\times\boldsymbol{H_r}=\left(\frac{\omega}{c}\right)^2\boldsymbol{H_r} \tag{2.20}$$

となる。これが、磁場の空間の偏微分連立方程式で、境界条件を課して解くことになる。その解は磁場の空間分布を表す。これと、式 (2.17) を掛けあわせたものが実際の電磁場の状態を表す。

この方程式の解は波になっており、電磁場は複素数で書かれるのが普通である。従って、式 (2.20) の $\boldsymbol{H_r}$ は複素数である。磁場を実数部と虚数部

$$\Re(\boldsymbol{H_r})=\boldsymbol{H}_r \tag{2.21}$$

$$\Im(\boldsymbol{H_r})=\boldsymbol{H}_i \tag{2.22}$$

とする。これらを使って、式 (2.20) を実数部と虚数部を分けた方程式にすると、

$$\nabla\times\nabla\times\boldsymbol{H}_r=\frac{\omega^2}{c^2}\boldsymbol{H}_r \tag{2.23}$$

$$\nabla\times\nabla\times\boldsymbol{H}_i=\frac{\omega^2}{c^2}\boldsymbol{H}_i \tag{2.24}$$

となる。式 (2.20) は、この 2 つの微分方程式を含んでいる。

定在波 (Standing Wave) の場合は、実数部のみを考える。磁場は実数とするのである。虚数部としても良いが、今までの慣習で実数部のみとする事になっている。

以上で、高周波電磁場の磁場が満たすべき方程式を示した。磁場と全く同じ方法で、電場が表す方程式を計算できる。それは

$$\nabla\times\nabla\times\boldsymbol{E_r}=\left(\frac{\omega}{c}\right)^2\boldsymbol{E_r} \tag{2.25}$$

となる。

電磁場分布を表す方程式は、磁場を表す式 (2.20) と電場を表す式 (2.25) がある。それぞれは、独立ではなくマクスウェルの方程式の式 (2.3) や (2.4) で関連づけられている。境界条件を考え計算しやすい方の場を求め、もう一方の場はマクスウェルの方程式に代入 (微分) することにより計算することになる。

2.2 汎関数

磁場の空間分布を示す式 (2.20) の汎関数を示す。

式 (2.20) の汎関数は

$$F[\boldsymbol{H}]=\int\left[(\nabla\times\boldsymbol{H})\cdot(\nabla\times\boldsymbol{H}^*)-\left(\frac{\omega}{c}\right)^2\boldsymbol{H}\cdot\boldsymbol{H}^*\right]dV \tag{2.26}$$

である。ここで、$\boldsymbol{H}$ は磁場の空間分布である。式 (2.20) では $\boldsymbol{H_r}$ としていたが、簡潔に記述するために、添え字の $\boldsymbol{r}$ を省くことにする。また、アスタリスク $*$ は複素共役 (complex conjugate) を

表す。このように複素共役を使うと、汎関数が実数になり、計算が簡単になる。また、汎関数はエネルギーに関係していることが多く、このようにすると磁場のエネルギーに関係した量になる。

それでは、この式の第 1 変分がゼロになる条件が式 (2.20) を満足するかどうか調べる。第一変分は、$\boldsymbol{H}$ を $\delta\boldsymbol{H}$ 変化させたときの微小変化量で

$$
\begin{aligned}
\delta F &= F[\boldsymbol{H}+\delta\boldsymbol{H}] - F[\boldsymbol{H}] \\
&= \int \left[\{\nabla\times(\boldsymbol{H}+\delta\boldsymbol{H})\}\cdot\{\nabla\times(\boldsymbol{H}^*+\delta\boldsymbol{H}^*)\} - \left(\frac{\omega}{c}\right)^2 (\boldsymbol{H}+\delta\boldsymbol{H})\cdot(\boldsymbol{H}^*+\delta\boldsymbol{H}^*) \right] dV \\
&\qquad - \int \left[(\nabla\times\boldsymbol{H})\cdot(\nabla\times\boldsymbol{H}^*) - \left(\frac{\omega}{c}\right)^2 \boldsymbol{H}\cdot\boldsymbol{H}^* \right] dV \\
&\qquad 2\text{ 次の微少量を無視すると} \\
&= \int \left[(\nabla\times\boldsymbol{H})\cdot(\nabla\times\delta\boldsymbol{H}^*) + (\nabla\times\boldsymbol{H}^*)\cdot(\nabla\times\delta\boldsymbol{H}) - \left(\frac{\omega}{c}\right)^2 \{\boldsymbol{H}\cdot\delta\boldsymbol{H}^* + \boldsymbol{H}^*\cdot\delta\boldsymbol{H}\} \right] dV \\
&\qquad \text{ベクトル恒等式 } \nabla\cdot(\boldsymbol{V}\times\boldsymbol{W}) = \boldsymbol{W}\cdot(\nabla\times\boldsymbol{V}) - \boldsymbol{V}\cdot(\nabla\times\boldsymbol{W}) \text{ を使う} \\
&\qquad \boldsymbol{V} = (\nabla\times\boldsymbol{H}) \text{ あるいは } (\nabla\times\boldsymbol{H}^*), \quad \boldsymbol{W} = \delta\boldsymbol{H}^* \text{ あるいは } \delta\boldsymbol{H} \text{ とする。} \\
&= \int \left[-\nabla\cdot\{(\nabla\times\boldsymbol{H})\times\delta\boldsymbol{H}^*\} + \delta\boldsymbol{H}^*\cdot\{\nabla\times(\nabla\times\boldsymbol{H})\} - \left(\frac{\omega}{c}\right)^2 (\boldsymbol{H}\cdot\delta\boldsymbol{H}^*) \right] dV \\
&\qquad + \int \left[-\nabla\cdot\{(\nabla\times\boldsymbol{H}^*)\times\delta\boldsymbol{H}\} + \delta\boldsymbol{H}\cdot\{\nabla\times(\nabla\times\boldsymbol{H}^*)\} - \left(\frac{\omega}{c}\right)^2 (\boldsymbol{H}^*\cdot\delta\boldsymbol{H}) \right] dV \\
&\qquad \text{この式に発散定理を使い、式を整理すると} \\
&= -\int [(\nabla\times\boldsymbol{H})\times\delta\boldsymbol{H}^* + (\nabla\times\boldsymbol{H}^*)\times\delta\boldsymbol{H}]\cdot\boldsymbol{n}dS + \\
&\qquad \int \left[\left\{ \nabla\times\nabla\times\boldsymbol{H} - \left(\frac{\omega}{c}\right)^2 \boldsymbol{H} \right\}\cdot\delta\boldsymbol{H}^* + \left\{ \nabla\times\nabla\times\boldsymbol{H}^* - \left(\frac{\omega}{c}\right)^2 \boldsymbol{H}^* \right\}\cdot\delta\boldsymbol{H} \right] dV
\end{aligned}
\tag{2.27}
$$

となる。

ここで、任意の $\delta\boldsymbol{H}$ に対して、この第一変分 δF がゼロになる条件を考える。しかし、今回の関数は複素数になっている。第 1 変分 δF は実数であるが、$\boldsymbol{H}$ や $\delta\boldsymbol{H}$ は複素数である。この複素数の実数部と虚数部の変化に対して、第 1 変分がゼロとならなくてはならない。わかりやすくするために、複素数になっている部分を

$$\boldsymbol{H} = \boldsymbol{H}_r + i\boldsymbol{H}_i \tag{2.28}$$

$$\delta\boldsymbol{H} = \delta\boldsymbol{H}_r + i\delta\boldsymbol{H}_i \tag{2.29}$$

と実数部と虚数部に分ける。これらを、式 (2.27) に代入すると、

$$
\begin{aligned}
\delta F = &-2\int [(\nabla\times\boldsymbol{H}_r)\times\delta\boldsymbol{H}_r + (\nabla\times\boldsymbol{H}_i)\times\delta\boldsymbol{H}_i]\cdot\boldsymbol{n}dS + \\
&2\int \left[\left\{ \nabla\times\nabla\times\boldsymbol{H}_r - \left(\frac{\omega}{c}\right)^2 \boldsymbol{H}_r \right\}\cdot\delta\boldsymbol{H}_r + \left\{ \nabla\times\nabla\times\boldsymbol{H}_i - \left(\frac{\omega}{c}\right)^2 \boldsymbol{H}_i \right\}\cdot\delta\boldsymbol{H}_i \right] dV
\end{aligned}
\tag{2.30}
$$

となる。これが、実数部と虚数部に分けた汎関数の第 1 変分である。もちろん、任意の $\delta\boldsymbol{H}$ に対して、これがゼロになる条件を考えるのである。任意の $\delta\boldsymbol{H}$ と言うことは、任意の $\delta\boldsymbol{H}_r$ と $\delta\boldsymbol{H}_i$ に対して、第 1 変分がゼロになる条件を探すのである。

そのためには、この式の右辺第 1 項と 2 項がともにゼロにならなくてはならない。右辺第 1 項は、境界条件を表し、

$$\begin{cases} (\nabla\times\boldsymbol{H}_r)\times\boldsymbol{n}=0 \quad \text{または} \qquad \delta\boldsymbol{H}_r=0 \\ \qquad\qquad \text{かつ} \\ (\nabla\times\boldsymbol{H}_i)\times\boldsymbol{n}=0 \quad \text{または} \qquad \delta\boldsymbol{H}_i=0 \end{cases} \tag{2.31}$$

の場合、ゼロとなる。通常は、$(\nabla\times H)\times\boldsymbol{n}=0$ とする。これが自然境界条件で、ノイマン条件となる。この磁場の回転は、式 (2.4) より、$\nabla\times H=i\omega\varepsilon_0\boldsymbol{E}$ となる。従って、ノイマン条件は、$\boldsymbol{E}\times\boldsymbol{n}$ と書き直すことができる。すなわち、電場と境界の法線方向が一致するのである。これは、金属の境界条件である。すなわち、境界を指定しなければ、自然に金属の境界条件が満足されるのである。一方、$\delta\boldsymbol{H}=0$ はディリクレ条件で、境界の値を指定した場合である。

第 2 項がゼロとなるのは、

$$\begin{cases} \nabla\times\nabla\times\boldsymbol{H}_r-\left(\dfrac{\omega}{c}\right)^2\boldsymbol{H}_r=0 \\ \qquad\qquad \text{かつ} \\ \nabla\times\nabla\times\boldsymbol{H}_i-\left(\dfrac{\omega}{c}\right)^2\boldsymbol{H}_i=0 \end{cases} \tag{2.32}$$

となる必要がある。これは、マクスウェルの方程式から導かれた磁場の偏微分方程式 (2.25) と全く同等である。

以上のことから、高周波の電磁場の磁場を計算するためには、式 (2.26) の第一変分をゼロにすればよいことが分かる。静磁場のマクスウェルの方程式は、式 (2.26) の第 1 変分をゼロにするのと等しいのである。

ここでは、軸対称空洞内部の電磁場を求めるための汎関数を示す。問題は定在波に限るものとする。

軸対称問題では通常、円柱座標系を使う。この場合、空洞の形状は完全軸対称である。定在波の場合、磁場は実数として取り扱うことができる。式 (2.26) では円柱座標系の回転の演算が表れ、それは

$$\nabla\times\boldsymbol{H}=\left[\frac{1}{r}\frac{\partial H_z}{\partial\theta}-\frac{\partial H_\theta}{\partial z}\right]\hat{\boldsymbol{r}}+\left[\frac{\partial H_r}{\partial z}-\frac{\partial H_z}{\partial r}\right]\hat{\boldsymbol{\theta}}+\frac{1}{r}\left[\frac{\partial}{\partial r}(rH_\theta)-\frac{\partial H_r}{\partial\theta}\right]\hat{\boldsymbol{z}} \tag{2.33}$$

である。

一般には、これを、汎関数の式 (2.26) に代入することになる。しかし、通常の空洞では最も共振周波数の低いモードが重要になる。これが、運転モードとなり、真っ先に解析したいモードである。このモードは、場が $\hat{\boldsymbol{\theta}}$ 方向の依存性を持たず、磁場は H_θ のみである。このモードの磁場の回転は、

$$\begin{aligned} \nabla\times\boldsymbol{H} &= \left[-\frac{\partial H_\theta}{\partial z}\right]\hat{\boldsymbol{r}}+\frac{1}{r}\left[\frac{\partial}{\partial r}(rH_\theta)\right]\hat{\boldsymbol{z}} \\ &= \left(-\frac{\partial H_\theta}{\partial z}\right)\hat{\boldsymbol{r}}+\left(\frac{H_\theta}{r}+\frac{\partial H_\theta}{\partial r}\right)\hat{\boldsymbol{z}} \end{aligned} \tag{2.34}$$

となる。この回転の結果を汎関数の式 (2.26) に適用すると、

$$
\begin{aligned}
F[H_\theta] &= \int \left[\left(\frac{\partial H_\theta}{\partial z} \right)^2 + \left(\frac{H_\theta}{r} + \frac{\partial H_\theta}{\partial r} \right)^2 - \left(\frac{\omega}{c} \right)^2 H_\theta^2 \right] dV \\
&= \int \left[\left(\frac{\partial H_\theta}{\partial z} \right)^2 + \left(\frac{\partial H_\theta}{\partial r} \right)^2 + 2 \frac{H_\theta}{r} \frac{\partial H_\theta}{\partial r} + \left(\frac{H_\theta}{r} \right)^2 - \left(\frac{\omega}{c} \right)^2 H_\theta^2 \right] 2\pi r dr dz
\end{aligned} \tag{2.35}
$$

となる。

2.3 離散化

ヘルムホルツ方程式を有限要素法によって固有値問題にすると、

$$
\boldsymbol{K}\boldsymbol{u} = \lambda \boldsymbol{M}\boldsymbol{u} \tag{2.36}
$$

という形の一般化固有値問題になる。ここで、ディレクレ条件は斉次ディレクレ条件とし、節点の番号は、ディレクレ条件が最後になっているものとする。すなわち、ディレクレ条件以外の節点が 1 番から m 番まで、$m+1$ 番から n 番までは、斉次ディレクレ条件とする。このような前提条件で、m 行 m 列の行列を以下のようにしてつくる。

少し、(2.35) 式を変形する。$\lambda = \left(\frac{\omega}{c} \right)^2$、$u = H_\theta$ とおく。また、2π は定数なので省略する。

$$
J = \iint_D r \left\{ \left(\frac{\partial u}{\partial r} \right)^2 + \left(\frac{\partial u}{\partial z} \right)^2 \right\} dr dz + 2 \iint_D u \frac{\partial u}{\partial r} dr dz + \iint_D \frac{u^2}{r} dr dz - \lambda \iint_D r u^2 dr dz \tag{2.37}
$$

この式の積分領域を、三角形 1 次要素に分割して計算する。ここで、節点番号 i、j、k を持つ三角形において、番号 i の点での u を u_i と表すことにする。このようにして、上の汎関数の微小変化（変分）が 0 となる解を計算する。

式 (2.37) の変分は、全三角形要素（(I)、(J)、$\cdots$）の和で次のように表すことができる。

$$
\begin{aligned}
\frac{\partial J[u]}{\partial u_i} &= \frac{\partial}{\partial u_i} \left[\iint_{(I)} r \left\{ \left(\frac{\partial u}{\partial r} \right)^2 + \left(\frac{\partial u}{\partial z} \right)^2 \right\} dr dz + 2 \iint_{(I)} u \frac{\partial u}{\partial r} dr dz + \iint_{(I)} \frac{u^2}{r} dr dz - \lambda \iint_{(I)} r u^2 dr dz \right] \\
&\quad + \frac{\partial}{\partial u_i} \left[\iint_{(J)} r \left\{ \left(\frac{\partial u}{\partial r} \right)^2 + \left(\frac{\partial u}{\partial z} \right)^2 \right\} dr dz + 2 \iint_{(J)} u \frac{\partial u}{\partial r} dr dz + \iint_{(J)} \frac{u^2}{r} dr dz - \lambda \iint_{(J)} r u^2 dr dz \right] \\
&\quad + \cdots
\end{aligned} \tag{2.38}
$$

特に、要素 (I) の寄与による項のみを書き出す。

$$
\left. \frac{\partial J[u]}{\partial u_i} \right|_{(I)} = \frac{\partial}{\partial u_i} \left[\iint_{(I)} r \left\{ \left(\frac{\partial u}{\partial r} \right)^2 + \left(\frac{\partial u}{\partial z} \right)^2 \right\} dr dz + 2 \iint_{(I)} u \frac{\partial u}{\partial r} dr dz + \iint_{(I)} \frac{u^2}{r} dr dz - \lambda \iint_{(I)} r u^2 dr dz \right] \tag{2.39}
$$

上式の積分を一つ一つ計算していく。

$$\frac{\partial}{\partial u_i}\iint_{(I)} r\left\{\left(\frac{\partial u}{\partial r}\right)^2+\left(\frac{\partial u}{\partial z}\right)^2\right\}drdz = 2\iint_{(I)} r\,drdz\left(\frac{\partial\phi_i}{\partial r}\frac{\partial\phi_i}{\partial r}+\frac{\partial\phi_i}{\partial z}\frac{\partial\phi_i}{\partial z}\right)u_i + 2\iint_{(I)} r\,drdz\left(\frac{\partial\phi_i}{\partial r}\frac{\partial\phi_j}{\partial r}+\frac{\partial\phi_j}{\partial z}\frac{\partial\phi_i}{\partial z}\right)u_j + 2\iint_{(I)} r\,drdz\left(\frac{\partial\phi_i}{\partial r}\frac{\partial\phi_k}{\partial r}+\frac{\partial\phi_k}{\partial z}\frac{\partial\phi_i}{\partial z}\right)u_k \tag{2.40}$$

$$\frac{\partial}{\partial u_i}\iint_{(I)} u\frac{\partial u}{\partial r}drdz = \iint_{(I)}\left(\phi_i\frac{\partial\phi_i}{\partial r}+\phi_i\frac{\partial\phi_i}{\partial r}\right)drdz\,u_i + \iint_{(I)}\left(\phi_i\frac{\partial\phi_j}{\partial r}+\phi_j\frac{\partial\phi_i}{\partial r}\right)drdz\,u_j + \iint_{(I)}\left(\phi_i\frac{\partial\phi_k}{\partial r}+\phi_k\frac{\partial\phi_i}{\partial r}\right)drdz\,u_k \tag{2.41}$$

$$\frac{\partial}{\partial u_i}\iint_{(I)}\frac{u^2}{r}drdz = 2\iint_{(I)}\frac{1}{r}\phi_i\phi_i drdz\,u_i + 2\iint_{(I)}\frac{1}{r}\phi_i\phi_j drdz\,u_j + 2\iint_{(I)}\frac{1}{r}\phi_i\phi_k drdz\,u_k \tag{2.42}$$

$$\frac{\partial}{\partial u_i}\iint_{(I)} ru^2 drdz = 2\iint_{(I)} r\phi_i\phi_i drdz\,u_i = 2\iint_{(I)} r\phi_i\phi_j drdz\,u_j = 2\iint_{(I)} r\phi_i\phi_k drdz\,u_k \tag{2.43}$$

これを元に行列の要素を決めると以下のようになる。

$$k_{ij}|_{(I)} = \left(\frac{\partial\phi_i}{\partial r}\frac{\partial\phi_j}{\partial r}+\frac{\partial\phi_i}{\partial z}\frac{\partial\phi_j}{\partial z}\right)\iint_{(I)} r\,drdz + \iint_{(I)}\left(\phi_i\frac{\partial\phi_j}{\partial r}+\phi_j\frac{\partial\phi_i}{\partial r}\right)drdz + \iint_{(I)}\frac{1}{r}\phi_i\phi_j drdz \tag{2.44}$$

$$m_{ij}|_{(I)} = \iint_{(I)} r\phi_i\phi_j drdz \tag{2.45}$$

ここで、ϕ は三角形内部を 1 次近似するための関数で、形状関数と呼ばれるものである。詳細は戸川隼人氏の「変分法と有限要素法」 [1] を参考にしてもらいたい。

$$\phi_i = \frac{1}{\Delta}\begin{vmatrix}1 & z & r\\ 1 & z_j & r_j\\ 1 & z_k & r_k\end{vmatrix} \tag{2.46}$$

Δ は三角形の面積の 2 倍で、

$$\Delta = r_i z_j + r_j z_k + r_k z_i - z_i r_j - z_j r_k - z_k r_i \tag{2.47}$$

となる。さらに積分を計算する。これらの式の積分範囲は三角形要素の内部である。例として図 2.1 のような三角形で考える。三角形の各辺を 1 次関数 f_a、f_b、f_c で表すと、次のように書ける。

$$f_a = \frac{r_k - r_j}{z_k - z_j}(z - z_j) + r_j \tag{2.48}$$

$$f_b = \frac{r_i - r_j}{z_i - z_j}(z - z_j) + r_j \tag{2.49}$$

$$f_c = \frac{r_k - r_i}{z_k - z_i}(z - z_i) + r_i \tag{2.50}$$

よって、三角形内部の積分範囲は次のようになる。

$$\iint drdz = \int_{z_j}^{z_i}\int_{f_a}^{f_b} drdz + \int_{z_i}^{z_k}\int_{f_a}^{f_c} drdz \tag{2.51}$$

これにしたがって、積分を計算していく。

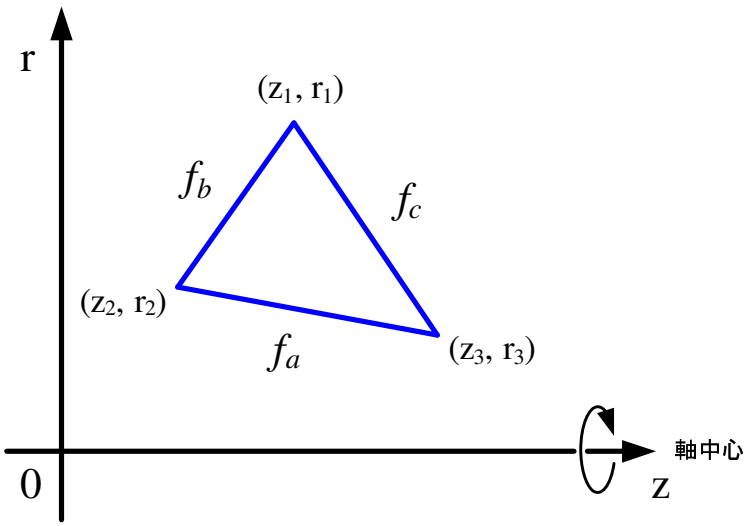

図 2.1: 積分する三角形領域

$$\iint_{(I)} r\,drdz\left(\frac{\partial\phi_i}{\partial r}\frac{\partial\phi_i}{\partial r}+\frac{\partial\phi_i}{\partial z}\frac{\partial\phi_i}{\partial z}\right)=\frac{1}{6\Delta}(r_i+r_j+r_k)[(z_j-z_k)(z_k-z_i)+(r_j-r_k)(r_k-r_i)] \tag{2.52}$$

$$\iint_{(I)}\left(\phi_i\frac{\partial\phi_j}{\partial r}+\phi_j\frac{\partial\phi_i}{\partial r}\right)drdz=\frac{z_j-z_k}{6}+\frac{z_k-z_i}{6} \tag{2.53}$$

次の計算は少し近似の計算をする。 r_0 は三角形の重心座標なので $r_0=(r_i+r_j+r_k)/3$ である。

$$\begin{aligned}\iint_{(I)}\frac{1}{r}\phi_i\phi_i drdz &= \frac{1}{r_0}\iint_{(I)}\phi_i\phi_i drdz \\ &= \begin{cases}\dfrac{1}{r_0}\dfrac{1}{24\Delta} & (i\neq j)\\[2ex] \dfrac{1}{r_0}\dfrac{1}{12\Delta} & (i=j)\end{cases}\end{aligned} \tag{2.54}$$

$$\iint_{(I)} r\phi_i\phi_j drdz=\begin{cases}\dfrac{1}{120}(2r_i+2r_j+r_k)\Delta & (i\neq j)\\[2ex] \dfrac{1}{60}(3r_i+r_j+r_k)\Delta & (i=j)\end{cases} \tag{2.55}$$

以上の計算より、行列 $\boldsymbol{K}$ および $\boldsymbol{M}$ は

$$\begin{aligned} k_{ii}|_{(I)} &= \frac{1}{6\Delta}(r_i + r_j + r_k)[(z_j - z_k)^2 + (r_j - r_k)^2] \\ &\quad + \frac{z_j - z_k}{3} \\ &\quad + \frac{1}{r_0}\frac{1}{12\Delta} \end{aligned} \tag{2.56}$$

$$\begin{aligned} k_{ij}|_{(I)} &= \frac{1}{6\Delta}(r_i + r_j + r_k)[(z_j - z_k)(z_k - z_i) + (r_j - r_k)(r_k - r_i)] \\ &\quad + \frac{z_j - z_k}{6} + \frac{z_k - z_i}{6} \\ &\quad + \frac{1}{r_0}\frac{1}{24\Delta} \end{aligned} \tag{2.57}$$

$$m_{ii}|_{(I)} = \frac{1}{120}(2r_i + 2r_j + r_k)\Delta \tag{2.58}$$

$$m_{ij}|_{(I)} = \frac{1}{60}(3r_i + r_j + r_k)\Delta \tag{2.59}$$

として決定できる。

このようにして得られた行列から、固有値問題が与えられる。今回はこれを共役勾配法を用いて計算した。

2.4 電場の計算

マクスウェルの方程式を離散化して、磁場は求められた。しかし、磁場だけでなく電場も求める必要がある。電場は、磁場を微分すれば得られる。ここでは、離散的に求められた磁場の微分を計算する方法について述べる。

共振空洞の問題において、電場を求める式は次のように書ける。

$$\boldsymbol{E} = \frac{i}{\varepsilon_0 \omega} \nabla \times \boldsymbol{H} \tag{2.60}$$

電場 $\boldsymbol{E}$ は、磁場 $\boldsymbol{H}$ の回転を計算することによって得られる。

軸対称構造の空洞内を有限要素解析するにあたり、まず磁場を計算した。軸対称なので、磁場は θ 方向成分のみである。したがって、この磁場の回転から求められる電場は

$$E_r = -\frac{i}{\varepsilon_0 \omega}\frac{\partial H_\theta}{\partial z} \tag{2.61}$$

$$E_z = \frac{i}{\varepsilon_0 \omega}\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) \tag{2.62}$$

となる。

有限要素法では、図 2.2 に示すように、三角形領域の頂点で磁場 H_θ の値を計算する。これら 3 つの磁場から、要素内の電場 E_r と E_z を求めなくてはならない。まず初めに、要素内の任意の位置での磁場を計算する式を導く。今回の計算では要素内を 1 次で近似しているので、要素内の磁場は

$$H_\theta(z, r) = \alpha + \beta z + \gamma r \tag{2.63}$$

となる。この要素内の磁場の値を用いて、領域内の電場を計算する。式 (2.63) により式 (2.61) と (2.62) を評価すると

$$E_r = -\frac{i}{\varepsilon_0 \omega}\beta \tag{2.64}$$

$$E_z = \frac{i}{\varepsilon_0 \omega}\left(\frac{\alpha + \beta z}{r} + 2\gamma\right) \tag{2.65}$$

を導くことができる。これらの式中の α、β、γ を求めることができれば電場を計算することができる。

まず、三角形の 3 つの頂点で成り立つ次の関係を考える。

$$\begin{bmatrix} 1 & z_i & r_i \\ 1 & z_j & r_j \\ 1 & z_k & r_k \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} H_{\theta\, i} \\ H_{\theta\, j} \\ H_{\theta\, k} \end{bmatrix} \tag{2.66}$$

ここで、(z_i, r_i) と (z_j, r_j)、(z_k, r_k) は各頂点の座標、$H_{\theta\, i}$ と $H_{\theta\, j}$、$H_{\theta\, k}$ はその位置での磁場の強さである。この連立方程式の解は、

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 1 & z_i & r_i \\ 1 & z_j & r_j \\ 1 & z_k & r_k \end{bmatrix}^{-1} \begin{bmatrix} H_{\theta\, i} \\ H_{\theta\, j} \\ H_{\theta\, k} \end{bmatrix} \tag{2.67}$$

として計算できる。これで α、β、γ の各係数が求められた。

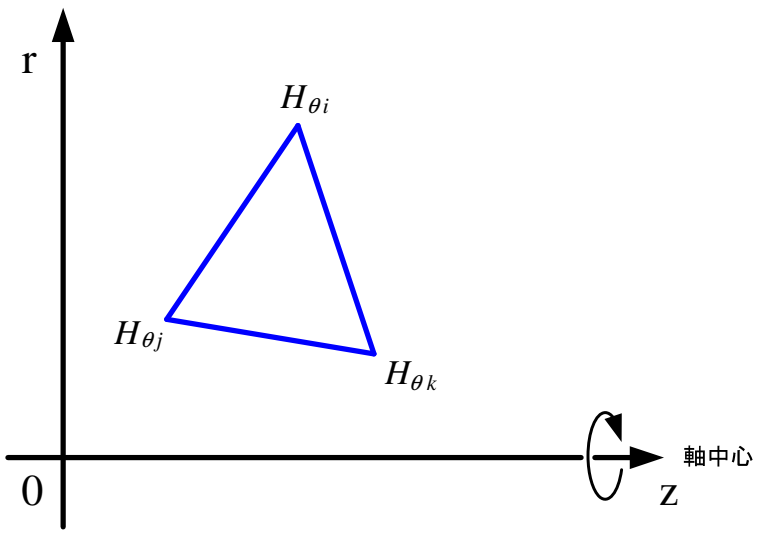

図 2.2: 三角形要素と各頂点での磁場

第3章　数値計算

3.1　プログラムの概要

今回の研究では、有限要素法を用いて軸対称の加速空洞内部の電磁場および共振周波数を求めるプログラムを作成した。使用した言語は、C++である。

プログラムは、大きく分けて次の 5 つの構成から成っている。

1. 領域中の各三角形の座標や境界条件を得る
2. 係数行列を決定する
3. 固有値問題を解き、固有値と磁場を求める
4. 磁場の回転を計算し、電場を求める
5. 計算結果を出力する

まず、本研究室で開発されたメッシュジェネレータから出力されるファイルを読み、計算領域の情報を得る。節点の座標とそれが境界条件かどうか、さらに三角形がどの節点を持っているかということがここでわかる。

そしてその情報をもとに係数行列を決定する。(2.56) 〜(2.59) 式を計算することでこれを行う。

係数行列が決まったら、これを用いて固有値問題を解き、固有値と磁場の分布が得られる。共振周波数は、固有値がわかれば次の関係

$$\left(\frac{\omega}{c}\right)^2 = \lambda \tag{3.1}$$

より

$$f = \frac{c}{2\pi}\sqrt{\lambda} \tag{3.2}$$

と求められる。c は光の速度（$c = 299792458$ [m/s]）である。

第4章　計算結果

本研究で作成したプログラムを用いて、KEK-PF の加速空洞 [2] を解析した。プログラムで計算を行い求められた共振周波数は、499.99MHz であった。実際に測定した値は 499.5MHz であったので、よい精度で計算できていることがわかる。

図 4.1: 空洞内の電磁場

上の図は、空洞内の電磁場を表している。色の分布が磁場の強さを示しており、白から黒へ変化するにつれて磁場が弱くなっている。また、横に伸びている黒い線は電気力線である。金属壁に対して垂直になっていることが観察できる。

第5章　考察

今回作成したプログラムの計算精度を確かめるために、図 5.1 の空洞を用いて精度の検証を行った。半径 1、高さ 1 の円柱型の共振空洞である。この空洞の固有値の解析解は簡単に求めることができ、その値は 5.783185963 である。

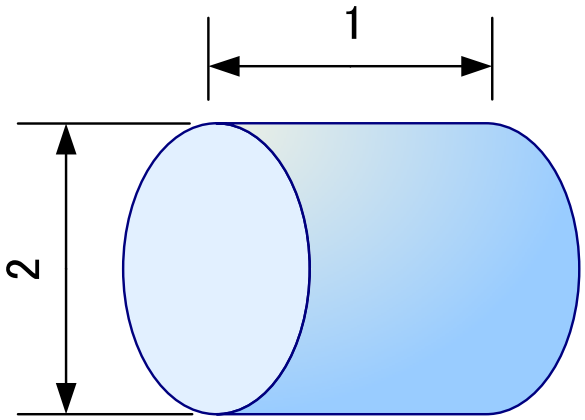

図 5.1: 円柱形の共振空洞

表 5.1 に、設定数と誤差の関係を示す。節点数を多くすればするほど誤差の値が小さくなっており、節点数の一番多いところで誤差は 10^{-4} 以下、つまり 0.01% 以下になっていることがわかる。また、節点数と誤差の関係をグラフ化したものを図 5.2 に示す。このグラフから、節点数を無限に増やすことができれば、誤差は 0 となることがわかる。しかし節点数の多さに応じて使用メモリ量が急激に増えるため、使用できる最大の節点数はコンピュータの性能に依存する。また、計算量が増えると丸め誤差の割合が大きくなってくるため、節点数を増やしすぎると誤差が大きくなる可能性がある。

表 5.1: 節点数と誤差との関係

節点数 $^{-1}$	節点数	固有値	誤差
0.000123	8125	5.78368	0.0000854
0.000127	7898	5.78373	0.0000941
0.000152	6562	5.78380	0.0001062
0.000180	5563	5.78391	0.0001252
0.000213	4703	5.78401	0.0001425
0.000254	3937	5.78416	0.0001684
0.000275	3634	5.78422	0.0001788
0.000549	1820	5.78529	0.0003638
0.000852	1174	5.78628	0.0005350

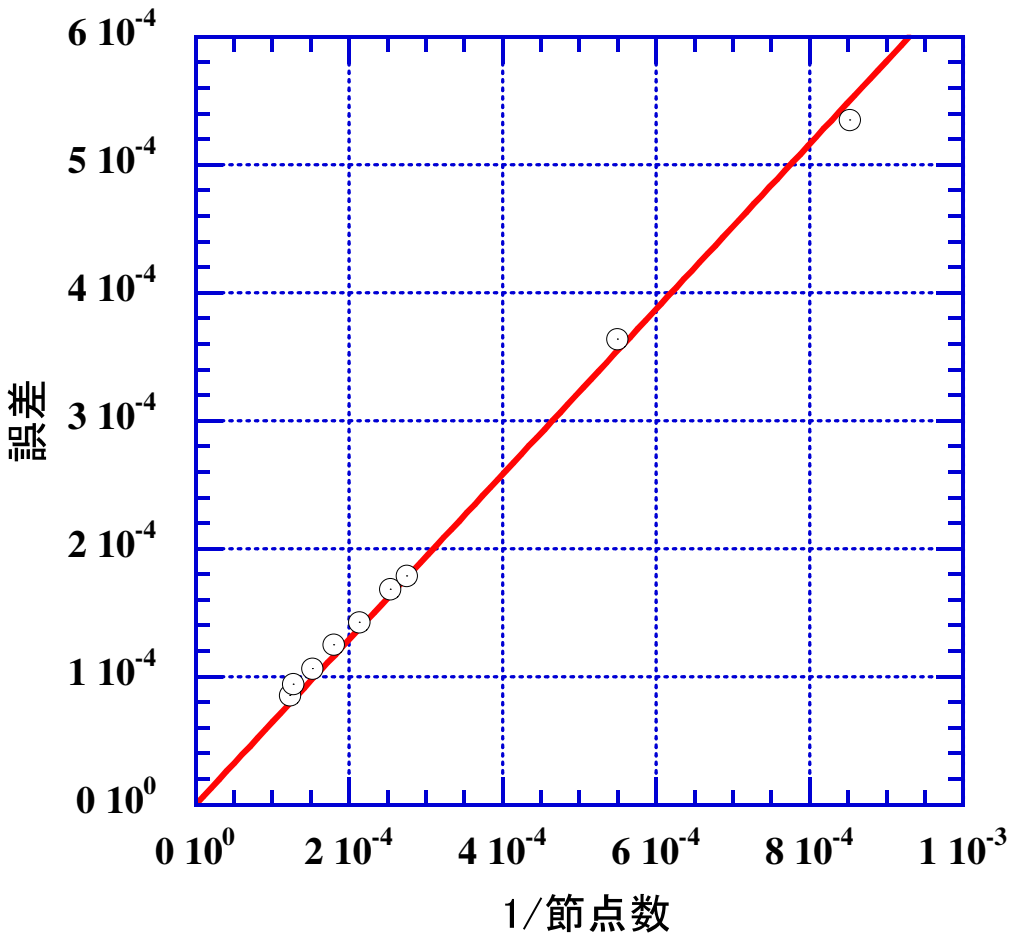

図 5.2: 節点数と誤差の関係

第6章　まとめ

1年間研究に取り組んだ結果として、計算精度が 10^{-4} 程度のプログラムをつくることができた。

今後の課題として、まず計算精度の向上が挙げられる。目標は、10^{-6} 程度である。そのためには、三角形要素の内部を2次近似することや、アダプティブメッシュを用いることなどが必要である。

また、進行波を取り扱うという課題も残っている。今回は定在波のみを扱っていたので、進行波問題も解けるようになればさらにプログラムの応用範囲が広がるであろう。

関連図書

[1] 戸川隼人. 変分法と有限要素法. 現代応用数学の基礎. 日本評論社, 1994.

[2] Yamazaki Yoshishige, Takata Koji, and Tokumoto Shuichi. Measurement of the longitudinal and transverse coupling impedances of the higher-order modes of the re-entrant accelerating cavity. *KEK 80-8*, Aug 1980.

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

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <algorithm>
#include <cmath>
#include "femapp.h"
#include "../eigen/matrix.h"
#include "../eigen/eigen_value.h"
using namespace std;


FemApp::FemApp()
{
}

FemApp::~FemApp()
{
}

void FemApp::set_factor_3()
{
        const int& N = NN;

        Matrix mtx_a_b(N, N);  // 一時的な行列 あとでサイズを減らす
        Matrix mtx_k_b(N, N);
        //mtx_a.change_size(N, N);
        //mtx_k.change_size(N, N);

        // 0で埋める
        //for (int i=1; i<=N; i++) {
        //      fill(mtx_a[i].begin(), mtx_a[i].end(), 0.0);
        //      fill(mtx_k[i].begin(), mtx_k[i].end(), 0.0);
        //}

        //cout << "factor_1" << endl;

        for (int i=1; i<=NE; i++) {
                // わかりやすいように別名をつける
                double& xi = elem[i-1].node[0]->x;
                double& xj = elem[i-1].node[1]->x;
                double& xk = elem[i-1].node[2]->x;
                double& yi = elem[i-1].node[0]->y;
                double& yj = elem[i-1].node[1]->y;
                double& yk = elem[i-1].node[2]->y;
                double delta = fabs((xj-xi)*(yk-yi) - (yj-yi)*(xk-xi));
                int I = elem[i-1].node[0]->index + 1;
                int J = elem[i-1].node[1]->index + 1;
                int K = elem[i-1].node[2]->index + 1;
```

```
        // * 左辺の行列 A
        // (1)
        double xijk = (xi+xj+xk)/(6*delta);
        mtx_a_b[I][I] += xijk * ((xk-xj)*(xk-xj) + (yj-yk)*(yj-yk));  // Ui で変分
        mtx_a_b[I][J] += xijk * ((xi-xk)*(xk-xj) + (yk-yi)*(yj-yk));
        mtx_a_b[I][K] += xijk * ((xj-xi)*(xk-xj) + (yi-yj)*(yj-yk));
        mtx_a_b[J][I] += xijk * ((xk-xj)*(xi-xk) + (yj-yk)*(yk-yi));  // Uj で変分
        mtx_a_b[J][J] += xijk * ((xi-xk)*(xi-xk) + (yk-yi)*(yk-yi));
        mtx_a_b[J][K] += xijk * ((xj-xi)*(xi-xk) + (yi-yj)*(yk-yi));
        mtx_a_b[K][I] += xijk * ((xk-xj)*(xj-xi) + (yj-yk)*(yi-yj));  // Uk で変分
        mtx_a_b[K][J] += xijk * ((xi-xk)*(xj-xi) + (yk-yi)*(yi-yj));
        mtx_a_b[K][K] += xijk * ((xj-xi)*(xj-xi) + (yi-yj)*(yi-yj));

        // (2)
        mtx_a_b[I][I] += -(yk-yj)/3.0;
        mtx_a_b[I][J] +=  (yj-yi)/6.0;
        mtx_a_b[I][K] +=  (yi-yk)/6.0;
        mtx_a_b[J][I] +=  (yj-yi)/6.0;
        mtx_a_b[J][J] += -(yi-yk)/3.0;
        mtx_a_b[J][K] +=  (yk-yj)/6.0;
        mtx_a_b[K][I] +=  (yi-yk)/6.0;
        mtx_a_b[K][J] +=  (yk-yj)/6.0;
        mtx_a_b[K][K] += -(yj-yi)/3.0;

        // (3)
        double coef = delta / ((xi+xj+xk)/3);
        mtx_a_b[I][I] += coef/12.0;
        mtx_a_b[I][J] += coef/24.0;
        mtx_a_b[I][K] += coef/24.0;
        mtx_a_b[J][I] += coef/24.0;
        mtx_a_b[J][J] += coef/12.0;
        mtx_a_b[J][K] += coef/24.0;
        mtx_a_b[K][I] += coef/24.0;
        mtx_a_b[K][J] += coef/24.0;
        mtx_a_b[K][K] += coef/12.0;

        // * 右辺の行列 K
        coef = delta/60.0;
        mtx_k_b[I][I] += coef * (3.0*xi + 1.0*xj + 1.0*xk);
        mtx_k_b[I][J] += coef * (1.0*xi + 1.0*xj + 0.5*xk);
        mtx_k_b[I][K] += coef * (1.0*xi + 0.5*xj + 1.0*xk);
        mtx_k_b[J][I] += coef * (1.0*xi + 1.0*xj + 0.5*xk);
        mtx_k_b[J][J] += coef * (1.0*xi + 3.0*xj + 1.0*xk);
        mtx_k_b[J][K] += coef * (0.5*xi + 1.0*xj + 1.0*xk);
        mtx_k_b[K][I] += coef * (1.0*xi + 0.5*xj + 1.0*xk);
        mtx_k_b[K][J] += coef * (0.5*xi + 1.0*xj + 1.0*xk);
        mtx_k_b[K][K] += coef * (1.0*xi + 1.0*xj + 3.0*xk);
}

// 行列のサイズを減らす
// 最大列数を求める
int col_max = 0;
const double ERR = 1e-10;

for (int i=1; i<=N; i++) {
        int col = 0;
        for (int j=1; j<=N; j++) {
                if (fabs(mtx_a_b[i][j]) > ERR) {   // 0 でない
                        col++;
                }
        }
```

```
                if (col > col_max) {
                        col_max = col;
                }
        }

        // コピーする
        mtx_a.change_size(N, col_max);
        mtx_k.change_size(N, col_max);
        mtx_idx.change_size(N, col_max);

        for (int i=1; i<=N; i++) {
                /*
                // まず 0 な列を探す
                int col_zero = 0;
                for (int col_zero=1; col_zero<=N; col_zero++) {
                        if (mtx_a_b[i][col_zero] < ERR) {
                                break;
                        }
                }
                */

                //
                int k = 1;
                for (int j=1; j<=N; j++) {
                        if (fabs(mtx_a_b[i][j]) > ERR) {
                                mtx_a[i][k] = mtx_a_b[i][j];
                                mtx_k[i][k] = mtx_k_b[i][j];
                                mtx_idx[i][k] = j;
                                k++;
                        }
                }
                /*
                for (int j=k; j<=col_max; j++) {
                        mtx_idx[i][j] = 0; //col_zero;
                }
                */
        }

}

// 固有値を返す
double FemApp::solve_eigen()
{
        vec_u.resize(NN+1);
        for (int i=1; i<=NN; i++) {
                if (node[i].is_bc) {
                        vec_u[i] = node[i].value;
                }
        }

        return eigen_value(mtx_a, mtx_k, mtx_idx, vec_u, arr_u);
}

void FemApp::calc_rotation()
{
        const double ERR = 1e-10;

        for (int i=0; i<NE; i++) {
                double& xi = elem[i].node[0]->x;
                double& xj = elem[i].node[1]->x;
                double& xk = elem[i].node[2]->x;
```

```
                double& yi = elem[i].node[0]->y;
                double& yj = elem[i].node[1]->y;
                double& yk = elem[i].node[2]->y;
                double delta = fabs((xj-xi)*(yk-yi) - (yj-yi)*(xk-xi));

                // value は解でない
                double beta  = ((yj-yk)*vec_u[elem[i].node[0]->index]
                        + (yk-yi)*vec_u[elem[i].node[1]->index]
                        + (yi-yj)*vec_u[elem[i].node[2]->index])/delta;
                double gamma = (-(xj-xk)*vec_u[elem[i].node[0]->index]
                        - (xk-xi)*vec_u[elem[i].node[1]->index]
                        - (xi-xj)*vec_u[elem[i].node[2]->index])/delta;
                double alpha = ((xj*yk-xk*yj)*vec_u[elem[i].node[0]->index]
                        + (xk*yi-xi*yk)*vec_u[elem[i].node[1]->index]
                        + (xi*yj-xj*yi)*vec_u[elem[i].node[2]->index])/delta;

                for (int j=0; j<3; j++) {
                        if (elem[i].node[j]->x < ERR) {
                                elem[i].node[j]->er.push_back(0);
                                elem[i].node[j]->ez.push_back(2*gamma);
                        }
                        else {
                                elem[i].node[j]->er.push_back(-beta);
                                elem[i].node[j]->ez.push_back(
                                        (alpha+beta*elem[i].node[j]->y)/elem[i].node[j]->x+2*gamma);
                        }
                }
        }

        // 平均
        for (int i=0; i<NN; i++) {
                double sumr = 0;
                double sumz = 0;
                for (int j=0; j<node[i].er.size(); j++) {
                        sumr += node[i].er[j];
                        sumz += node[i].ez[j];
                }
                node[i].er[0] = sumr / (double)node[i].er.size();
                node[i].ez[0] = sumz / (double)node[i].ez.size();
        }
}

// 解いた結果を出力
void FemApp::print_solution()
{
        ofstream fs("plot");
        for (int i=1; i<=NN; i++) {
                fs << left << showpoint << setfill('0') << setw(13) << setprecision(10)
                        << node[i-1].x << "\t";
                fs << left << showpoint << setfill('0') << setw(13) << setprecision(10)
                        << node[i-1].y << "\t";
                fs << left << setiosflags(ios::fixed) << showpoint << setfill('0') << setw(14) << setprecision(10)
                        << vec_u[i] << endl;
        }
        fs.close();
        fs.clear();

        fs.open("elems");
        for (int i=0; i<NE; i++) {
                for (int j=0; j<3; j++) {
                        fs << elem[i].node[j]->index << "\t";
```

```
                }
                fs << endl;
        }
}

void FemApp::print_solution2()
{
        ofstream fs((fname + ".field").c_str());

        // global
        /*
        fs << "$global" << endl;
        fs << "analysis_type=eigenmode" << endl;
        fs << "field_data=node Ht" << endl;
        fs << "$end" << endl;
        */

        // field
        fs << "$field" << endl;
        for (int i=0; i<NN; i++) {
                fs << i+1 << '\t';
                fs << vec_u[i+1] << '\t';
                fs << node[i].er[0] << '\t';
                fs << node[i].ez[0] << endl;
        }
        fs << "$end" << endl;
        fs << "$end_data" << endl;


        fs.close();

}

// xxxxxxx.meshout からメッシュの情報を読み込む
bool FemApp::read_mesh(const string filename)
{
        fname = filename;

        ifstream fs((filename + ".meshout").c_str());
        if (!fs) {
                cerr << "read_mesh() : cannot open " << filename << endl;
                return false;
        }

        char buf[1024];
        while (!fs.eof()) {
                fs.getline(buf, 1024);
                string str = buf;

                if (str[0] == '#' || str[0] == ' ') {
                        // コメント行
                        continue;
                }
                else if (str == "$global") {
                        // 節点数と要素数を読む
                        if (read_mesh_global(fs)) {
                                node.resize(NN);
                                elem.resize(NE);
                        }
                }
                else if (str == "$elements") {
```

```
                // 要素
                read_mesh_elements(fs);
            }
            else if (str == "$points") {
                // 節点
                read_mesh_points(fs);
            }
            else if (str == "$end_data") {
                break;
            }
        }

        fs.close();

        return true;
}

bool FemApp::read_mesh_global(ifstream& fs)
{
        char buf[1024];

        while (!fs.eof()) {
            fs.getline(buf, 1024);
            string str = buf;

            if (str[0] == '#' || str[0] == ' ') {
                continue;
            } else if (str == "$end") {
                return true;
            } else {
                if (str.find("node=") != -1) {
                    NN = atoi(str.substr(str.find("=")+1).c_str());
                } else if (str.find("nelm=") != -1) {
                    NE = atoi(str.substr(str.find("=")+1).c_str());
                }
            }
        }

        return true;
}
bool FemApp::read_mesh_elements(ifstream& fs)
{
        char buf[1024];
        int i = 0;

        while (!fs.eof()) {
            fs.getline(buf, 1024);
            string str = buf;

            if (str[0] == '#' || str[0] == ' ') {
                continue;
            } else if (str == "$end") {
                return true;
            } else {
                // str.substr(n)                    : str[0]～str[n-1]   を切り捨てる
                // str.substr(str.find("x")+1)  : str[0]～'x'          を切り捨てる
                // str.substr(0)                    : str
                // str.substr(0, n)                 : str[n]～'最後'     を切り捨てる
                // str.substr(0, str.find("x")) : str[n]～'x'          を切り捨てる

                elem[i].index = i;
```

```
                str = str.substr(str.find("\t")+1);  // 最初の数字を読み飛ばす
                // 1から始まる番号を，0から始まる番号に変えている（atoi(...)−1）
                elem[i].node[0] = &node[atoi(str.substr(0, str.find("\t")).c_str())−1];
                str = str.substr(str.find("\t")+1);
                elem[i].node[1] = &node[atoi(str.substr(0, str.find("\t")).c_str())−1];
                str = str.substr(str.find("\t")+1);
                elem[i].node[2] = &node[atoi(str.substr(0, str.find("\t")).c_str())−1];
                i++;
            }
        }

        return true;
}
bool FemApp::read_mesh_points(ifstream& fs)
{
        char buf[1024];
        int i = 0;

        while (!fs.eof()) {
            fs.getline(buf, 1024);
            string str = buf;

            if (str[0] == '#' || str[0] == ' ') {
                continue;
            } else if (str == "$end") {
                return true;
            } else {
                node[i].index = i;
                str = str.substr(str.find("\t")+1);  // 最初の数字を読み飛ばす
                node[i].x = atof(str.substr(0, str.find("\t")).c_str());
                str = str.substr(str.find("\t")+1);
                node[i].y = atof(str.substr(0, str.find("\t")).c_str());
                str = str.substr(str.find("\t")+1);
                if (str[0] == '1') {
                    // 境界条件
                    str = str.substr(str.find("\t")+1);
                    node[i].value = atof(str.c_str());
                    node[i].is_bc = true;
                }
                else {
                    node[i].value = 0.0;
                    node[i].is_bc = false;
                }
                i++;
            }
        }

        return true;
}
```