ロッド電極間に板状電極を挿入した
リニアイオントラップのイオン光学的研究

大阪大学理学研究科物理学専攻
安藤弘樹

平成 24 年 2 月 29 日

目 次

第1章　イントロダクション

近年では、質量分析法は、食品製造過程でのインライン分析、医療現場での血液分析、違法薬物の検知など、フィールド・サイエンスへの応用も期待されている。このような「現場での分析」では、装置が小型・軽量でポータブルであること、多くの夾雑物の中から目的の物質を正確に同定するために、高分解能であることが求められる。我々の研究室で開発されたマルチターン飛行時間型質量分析計 MULTUM[1][2] は、小型でありながら高い質量分解能を達成できるので、現場での分析に適している。

MULTUM は飛行時間型質量分析法を応用した質量分析装置である。飛行時間型質量分析法は 1946 年に Stephens により紹介された方法である [3]。静電場中で質量電荷比が異なるイオンを一定のエネルギーで加速し、その後、自由空間を飛行させた場合、イオンがある距離を進むのにかかる飛行時間は、質量電荷比が小さいイオンの方が短い。このように、静電場中でのイオンの飛行時間が質量に依存することを利用したのが飛行時間型質量分析法である。飛行時間型質量分析法ではイオンの飛行距離を長くするか、イオンを加速する際の初期位置の分布、運動量の分布による、同じ質量電荷比のイオンの検出器の位置での飛行時間のバラつきを小さくすることで分解能が向上する。MULTUM は同一飛行空間を複数回周回させることで長い飛行距離を稼ぎ高分解能を達成する。

また現場での分析ではサンプルに十分な前処理を施すことが難しいので、前処理が比較的容易な、大気圧下でサンプルをイオン化できる大気圧イオン源の一種であるバリア放電イオン源 [4] が有用である。この場合、大気圧イオン源と MULTUM を接続する方法について考慮する必要がある。飛行時間型質量分析計で質量分離を行うには、イオンをパケット化してパルス的に質量分離部に導入する必要がある。大気圧イオン源と飛行時間型質量分析計の接続では、連続的に生成するイオンをイオンの進行方向と直交する方向にパルス的に加速する、直交加速法が広く用いられている [5]。しかしながら、大気圧イオン源で生成したイオンを直交加速法により MULTUM に導入する場

合、イオンの利用効率が悪くなり微量分析に不利になるという問題が生じる。

そこで我々は、大気圧イオン源と MULTUM 接続のためのインターフェース、ロッド電極間に板状電極を挿入したリニアイオントラップ [6] [7] [8] を開発した。本装置は、リニアイオントラップ内部に生成したにイオンを蓄積し、蓄積したイオンを時間的・空間的に収束させながら排出できることが実験的に確認されている。本装置に、大気圧イオン源で生成したイオンを打ち込んで蓄積し、その後 MULTUM に向けて排出すれば、高いイオンの利用効率で MULTUM に時間的・空間的に収束させながら導入することができると考えられる。

本研究では、大気圧イオン源と MULTUM-S の接続のインターフェースとなる、ロッド電極間に板状電極を挿入したリニアイオントラップの性能向上を目的とする。そのためには、リニアイオントラップ内部のイオンの空間分布、運動状態を調べる必要がある。しかしながら、装置の電極形状を細かに変更しながら実験することは現実的ではなく、リニアイオントラップ内部のイオンの運動状態と空間分布を実験のみで見積もることは困難である。そこで、表面電荷法 [9][10] を用いた数値シミュレーションを行い、本装置におけるイオンの運動状態と空間分布について調べた。まず、大気圧イオン源で生成したイオンをリニアイオントラップに打ち込む過程をシミュレーションし、リニアトラップに蓄積されたイオンの空間分布・運動状態を見積もった。さらに、リニアイオントラップ内部に蓄積されたイオンを、リニアイオントラップの外部に排出する過程もシミュレーションし、MULTUM に導入するイオンの時間収束性・空間収束性を向上させる方法について考察した。また、ロッド電極間に挿入された板状電極の影響によりリニアイオントラップ内部の電場が乱れ、イオンの蓄積効率を低下させてしまう効果について定量的に評価した。さらに、板状電極のトラップ内部の電場への影響を、リニアイオントラップの電極形状を変更して相殺する方法について検討した。

本論文では、第２章ではシミュレーションの対象となる装置について述べ、第３章ではシミュレーションに用いた計算手法ついて述べる。そして、第４章以降ではシミュレーション結果について述べる。

第2章　本研究で想定する実験系

この章では、まず現在開発中の質量分析装置の全体像について明らかにする。次に一般的なリニアイオントラップについて概説する。そして、シミュレーションの対象となるロッド電極間に板状電極を挿入したリニアイオントラップについて説明する。

2.1　装置概要

我々の研究室では、小型でありながら高分解能が得られるマルチターン飛行時間型質量分析計 MULTUM-SⅡを開発した [11]。この装置は電源系、真空排気系を含めた大きさが 50.4[cm]×58.4[cm]×27.3[cm], 重量が 35[kg] と、持ち運びが可能なサイズでありながら、質量分解能は 30000 以上を達成できる。現在、この装置の特長を活かし、質量分析計を現場に持ち出して分析を行う、オンサイト・マススペクトロメトリーの展開が検討されている。現場での分析では、実験室のように十分なサンプルの前処理が難しく、多くの夾雑物の中から目的の物質を同定しなければならないという問題点がある。この問題を解決するために、我々の研究室では MULTUM-SⅡを質量分離部とした、現場での分析を視野に入れた質量分析装置を開発した。

MULTUM-SⅡは図 2.1 に示すような、４つの周回用の扇形電場と、入出射用の扇形電場から構成される飛行時間型質量分析計である。飛行時間型質量分析計の原理について簡単に述べる。図 2.2 のように、イオン源で一定の加速電圧 V で加速された、質量 m、価数 z のイオンが、距離 L を飛行した場合の飛行時間 T は式 (2.1) で与えられる。

$$T = L\sqrt{\frac{m}{2zeV}} \tag{2.1}$$

ここで e は素電荷である。したがって、飛行時間 T を測定することによって、イオンの質量電荷比 m/z を求めることができる。実際の装置では、イオン源でイオンをパルス電圧によって加速することでパルス状にし、検出器に到

達するイオンの強度と飛行時間の関係 (飛行時間スペクトル) を得る。質量分解能 $\frac{m}{\Delta m}$ は 、飛行時間スペクトルのピーク幅 ΔT を用いて式 (2.2) で与えられる。

$$\frac{m}{\Delta m} = \frac{T}{2\Delta T} \tag{2.2}$$

したがって、質量分解能を向上させるには、ピーク幅 ΔT を小さくするか、もしくは飛行距離を長くして飛行時間 T を延ばせばよい。MULTUM-SⅡの周回部のイオン光学系は、空間および飛行時間に関して完全収束を満たしている [12]。様々な初期条件をもったイオンが、周回部を多重周回しても、空間的・時間的に広がっていくことがない。式 (2.2) を用いて説明すれば、イオンは MULTUM-SⅡで多重周回することにより ΔT を一定に保ったまま飛行時間 T を増大されることになる。

MULTUM-SⅡへの入射時に大きな空間・角度・エネルギーの広がりを持ったイオンは電極に衝突して失われてしまい、確実に多重周回可能なイオンパケットのサイズは 0.5[mm]×2.0[mm] 程度であるといわれている。(図 2.3)。したがって、イオンパケットは MULTUM-SⅡの入射電極の位置で、0.5[mm]×2.0[mm] 程度に空間的に収束し (図 2.3)、かつ、質量分解能の向上の観点から検出器の位置で時間的に収束していることが要求される。

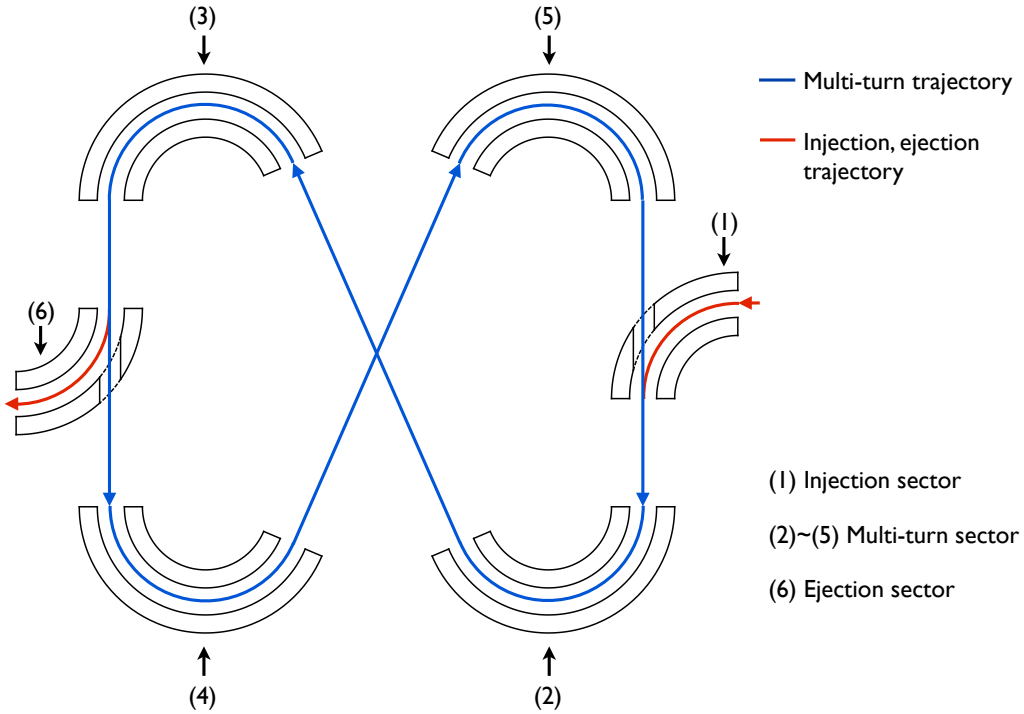


図 2.1: MULTUM-S の概念図

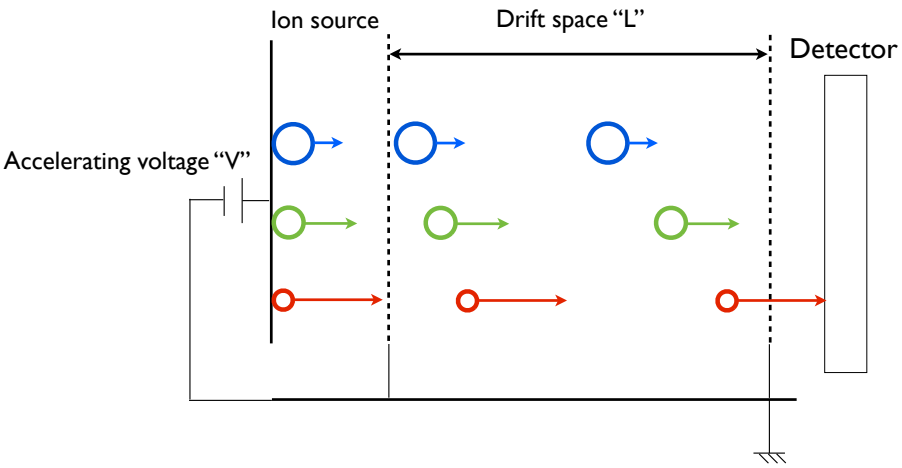


図 2.2: 飛行時間型質量分析法の模式図

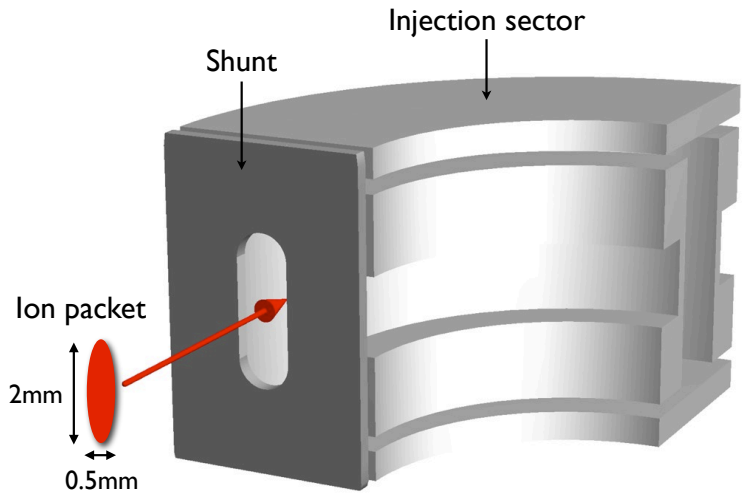


図 2.3: MULTUM-SⅡの入射電極と多重周回可能なイオンサイズ

また、現場で簡易な前処理のみでサンプルを測定するために、大気圧下でサンプルをイオン化できる、大気圧イオン源 [13] を用いるとよい。大気圧イオン源では、イオンは連続的に生成される。この場合、大気圧イオン源とMULTUM-SⅡの接続方法について考慮しなくてはならない。飛行時間型質量分析計による質量分離を行うには、イオンをパケット化してパルス的に質量分離部に導入することが求められる。この問題の解決法として、連続的に導入されるイオンを進行方向に直交する方向にパルス的に加速する直交加速法 [5] が一般的に用いられる (図 2.4)。しかしながら、大気圧イオン源で生成されたイオンを直交加速法により MULTUM-SⅡに導入した場合、MULTUM-SⅡで周回可能なイオンパケットのサイズが 0.5[mm]×2.0[mm] 程度と小さいので、イオンの利用効率が悪くなり、微量分析に不利になるという問題が生じる。そこで我々は、大気圧イオン源で生成したイオンを蓄積する機能と、蓄積したイオンを任意の位置で時間的・空間的に収束させながら排出する機能を有する、大気圧イオン源と MULTUM-SⅡ接続のための、ロッド電極間に板状電極を挿入したリニアイオントラップ [6][7] の開発に取り組んできた。

ロッド電極間に板状電極を挿入したリニアイオントラップの詳細については 2.3 で述べ、ここでは開発した質量分析装置全体の構成について説明する (図 2.5)。大気圧イオン源にて生成されたイオンは、いくつかの隔室を経て真空度を徐々に高くしていく差動排気系に導入される。差動排気系内部には、導入されたイオンを安定に輸送するためのイオンガイドが設けられており、イオンはイオンガイド 1 で 10^{-1}[Pa]、イオンガイド 2 で 10^{-2}[Pa] の真空中を輸送されていく。イオンガイド 2 を通過したイオンは、ロッド電極間に板状電極を挿入したリニアイオントラップ (2.3 参照) に導入される。イオンは、このリニアイオントラップ内部に一時的に蓄積され、その後、適当なタイミングで直交方向に排出される。さらにイオンは、時間収束させるために設けられた引き出し電極 1,2 により加速され、MULTUM-SⅡの入射電極の位置で時間収束する。リニアイオントラップと挿入電極 1,2 で、イオンを時間収束させる方法については 2.3 で述べる。MULTUM-SⅡに入射したイオンは質量分離されたのち、検出器に向けて出射され、リニアイオントラップから検出器までの飛行時間スペクトルを得る。

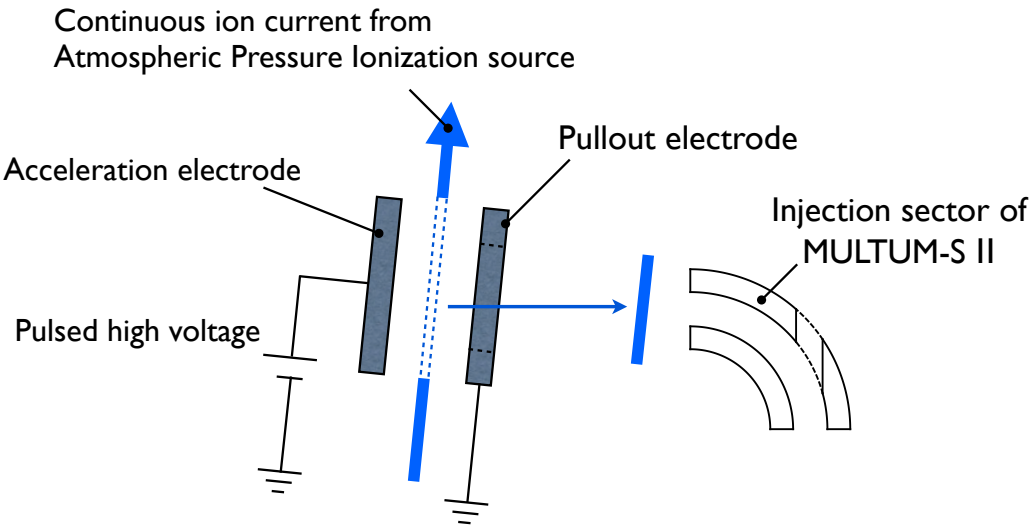


図 2.4: 直交加速法のによる MULTUM-SII へのイオンの導入

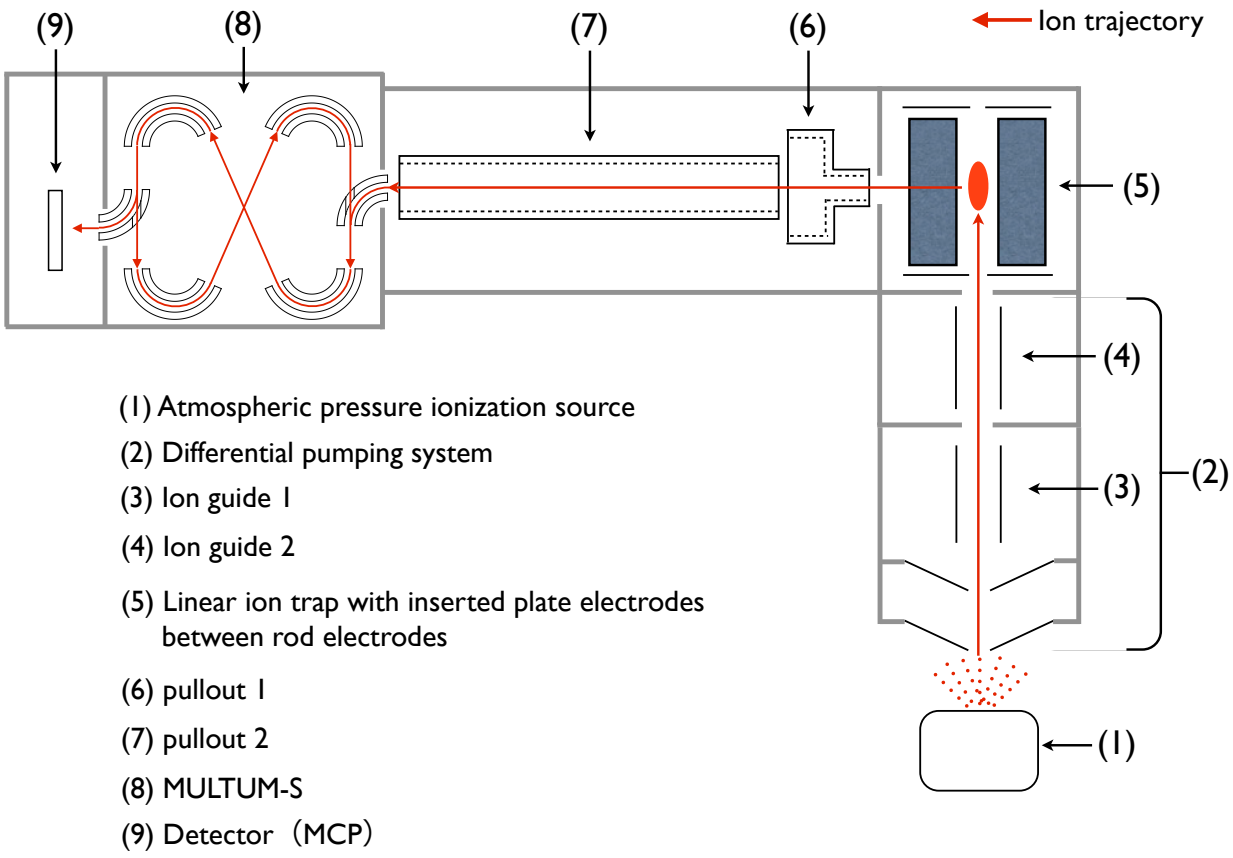


図 2.5: 装置全体図

2.2　リニアイオントラップ

ここでは一般的なリニアイオントラップについて概説する。リニアイオントラップとは図 2.6 に示すように、平行に配置された 4 本のロッド電極と、ロッド電極を両端から挟むように配置された 2 個のエンドキャップ電極から構成される。リニアイオントラップはロッド電極とエンドキャップに囲まれた空間に、イオンを 3 次元的に閉じ込めることができる。リニアイオントラップのイオン閉じ込めの原理について、まず $x-y$ 方向について説明し、つづいて z 方向について説明する（x, y, z の方向については図 2.6 を参照)。

4 本のロッド電極の断面は式 (2.3),(2.4) で表せる双曲線である。

$$x^2 - y^2 = r_0^2 \tag{2.3}$$

$$x^2 - y^2 = -r_0^2 \tag{2.4}$$

ここで r_0 は、式 (2.3),(2.4) で表される 4 本の双曲線の内接円半径である。図 2.6 のロッドの形状を $x-y$ 平面で見ると、図 2.7 のようになる。

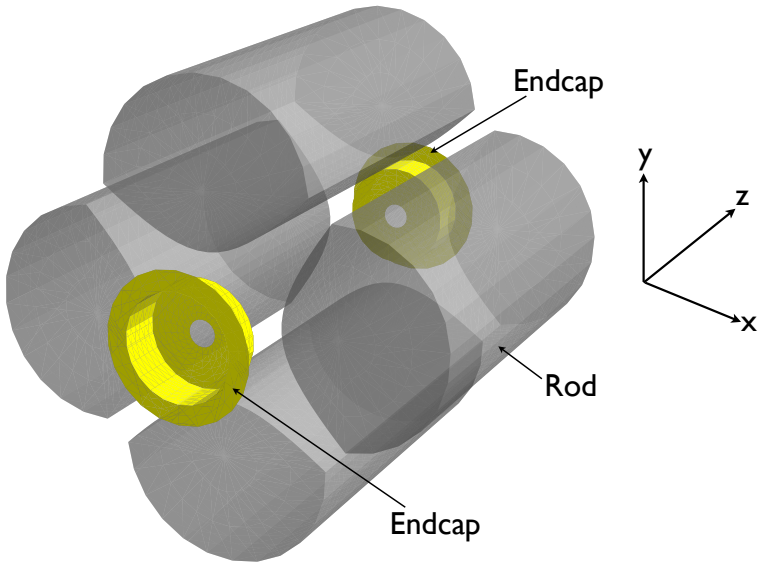


図 2.6: リニアイオントラップ

図 2.7 において、ϕ_+, ϕ_- はロッド電極に印加する電圧であり式 (2.5) で表される。

$$\phi_\pm = \pm\left(U + V cos\omega t\right) \tag{2.5}$$

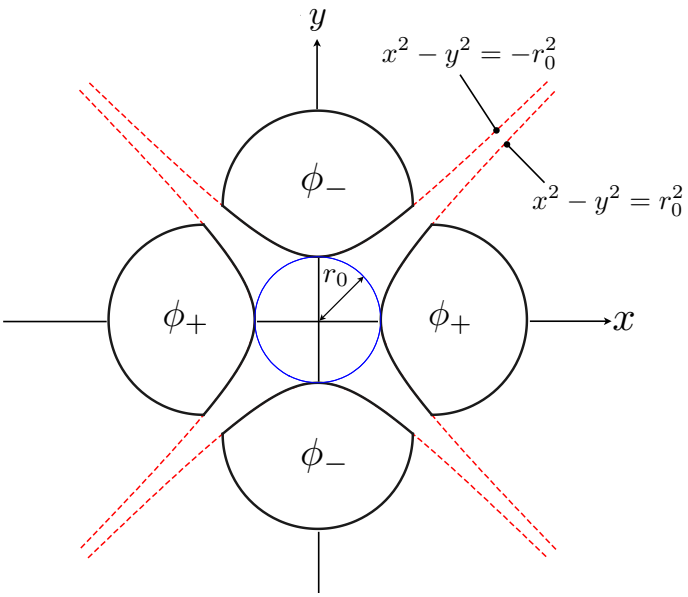


図 2.7: リニアイオントラップのロッド電極の形状と配置

ここで U はロッド電極に印加する直流成分の電圧、V はロッド電極に印加する交流成分の電圧の振幅である。ω はこの電圧の角周波数である。

図 2.7 のようにロッド電極に電圧を印加することで、x, y 方向にそれぞれ式 (2.6),(2.7) で表される電場が形成される。

$$E_x = -(U + Vcos\omega t)\frac{2x}{r_0^2} \tag{2.6}$$

$$E_y = (U + Vcos\omega t)\frac{2y}{r_0^2} \tag{2.7}$$

この電場中での質量 m、電荷 e のイオンの運動について説明する。式 (3.55)〜(2.10) の変数変換を行うと

$$\omega t = 2\xi \tag{2.8}$$

$$a_x = -a_y = \frac{8eU}{mr_0^2\omega^2} \tag{2.9}$$

$$q_x = -q_y = \frac{4eV}{mr_0^2\omega^2} \tag{2.10}$$

x, y 方向のイオンの運動方程式は同じ式で与えられ、式 (2.11) となる。

$$m\frac{d^2u}{d\xi^2} + (a_u + 2q_u cos2\xi)u = 0 \qquad (u = x, y) \tag{2.11}$$

この微分方程式は Mathieu 方程式と呼ばれ、一般解は式 (2.12) で与えられる。

$$u(\xi) = A \sum_{n=-\infty}^{\infty} C_{2n} cos(2n + \beta_u)\xi + B \sum_{n=-\infty}^{\infty} C_{2n} sin(2n + \beta_u)\xi \quad (2.12)$$

A と B は初期条件で決まり、C_{2n} は a_u, q_u の関数である。この方程式は a_u, q_u が、図 2.8 に示した $iso\text{-}\beta_x = 0.0,\ 1.0$ と $iso\text{-}\beta_y = 0.0,\ 1.0$ の曲線で囲まれる領域内にあれば、u の振幅は有限となり、イオンは $x - y$ 方向で閉じ込められる。イオンが $x - y$ 方向でトラップされる a_u, q_u 空間の領域を安定領域と呼ぶ。a_u, q_u が安定領域の外にあれば、u の振幅は発散し、イオンは $x - y$ 方向で閉じ込められない。

等しい β_u を与える点群は iso-β 線と呼ばれ、安定領域内では $0 \leq \beta_u \leq 1$ の範囲で変化する無次元の数である。イオンの $x - y$ 方向の軌道は β_u によって決定されるので、iso-β 線の導入はイオンの $x - y$ 方向の運動を理解するのに役立つ。

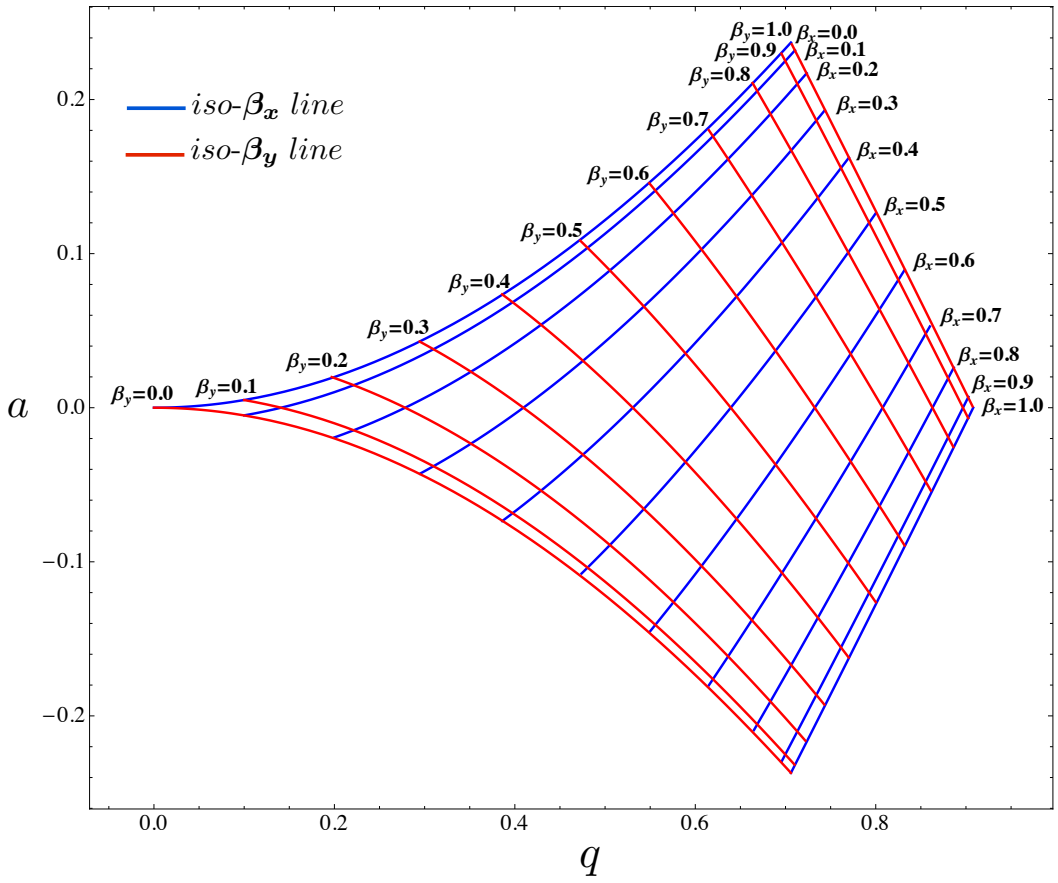
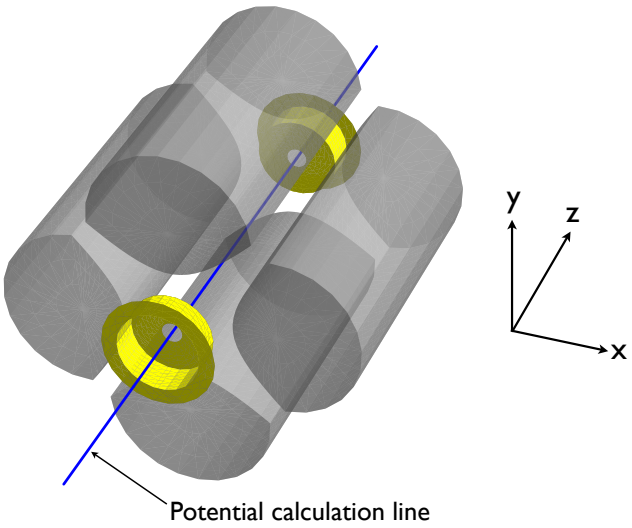
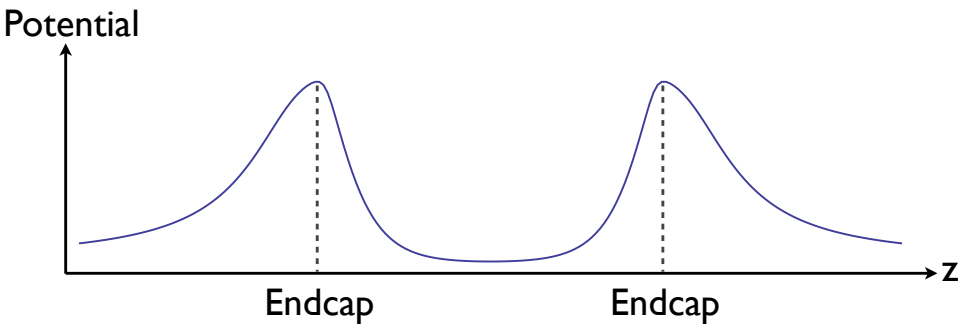


図 2.8: 安定領域と iso-β線

つづいて、リニアイオントラップの z 方向のイオンの閉じ込め原理について説明する（z 方向については図 2.6 を参照）。z 方向へのイオンの閉じ込めは、エンドキャップ電極に直流電圧を印加することで達成できる。エンドキャップ電極に直流電圧を印加したとき、図 2.9(a) に示すライン上に形成されるポテンシャルを図 2.9(b) に示す。直流電圧を印加したエンドキャップが形成するポテンシャルは、導体であるロッド電極の存在により弱められ、結果としてリニアイオントラップ内部には z 方向に井戸型のポテンシャルが形成される。この井戸型のポテンシャルにより、z 方向にもイオンを閉じ込めることができる。



(a) ポテンシャルを計算するライン



(b) エンドキャップに直流電圧を印加したときの (a) の計算ライン上でのポテンシャル

図 2.9: エンドキャップ電極により形成されるポテンシャル

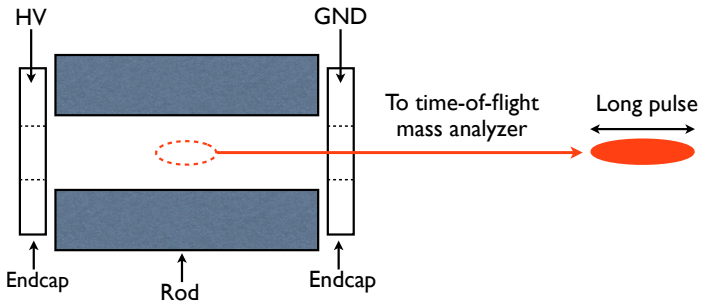
2.3 ロッド電極間に板状電極を挿入したリニアイオントラップ

リニアイオントラップに蓄積されたイオンを、外部に排出する方法のひとつとして、エンドキャップ電極にイオン排出用の電圧を印加する方法が考えられる (図 2.10(a))。この場合、2.2 でも説明したように、エンドキャップ電極に電圧を印加しても導体のロッド存在によって弱められてしまう。したがって、イオンを加速する電場が弱いため、初期エネルギーや初期位置などの違いによる飛行時間の広がりを収束することができないので、飛行時間型質量分析計への接続には適さない。

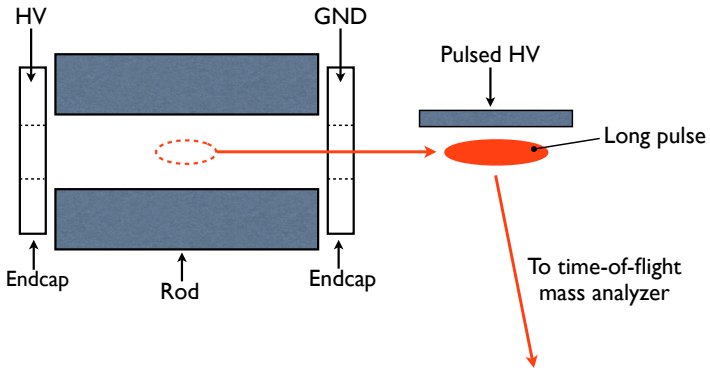
次に、エンドキャップ電極にイオン排出用の電圧を印加して、イオンをトラップ外部に排出した後に、直交加速法 [5] によりイオンをパルス化する方法が考えられる (図 2.10(b))。この方法は時間収束性には優れるが、トラップ外部に排出される際に質量電荷比の違いで長く広がってしまったイオンを、再び空間的に収束させることは困難である。良好な測定感度を目指すには、この方法は適切ではない。

先の 2 つの方法は、イオンをリニアイオントラップ外部に排出する際に、トラップ軸方向に空間的に広がってしまうことが問題であった。そこで、図 2.10(c) に示したような、トラップ軸と直交する方向にイオンを排出する方法について考える。まず、図 6.2(a) のように、ロッド電極のひとつにイオン排出用の穴を空け、ロッド電極にパルス電圧を印加する方法が考えられる。しかし、この方法では、穴の空いたロッドがトラップ電場を乱してしまう可能性が高い [14]。その上、ロッド電極に印加する電圧についても、イオン閉じ込め用の RF 電圧とイオン排出用のパルス電圧を速やかに切り替えることが求められ、電源系が複雑になる。別の方法として、ロッド電極の間隙からイオンを排出する方法が考えれる。図 7.2(b) は、リニアイオントラップの外部に板状電極を配置し、これらの電極に、リニアイオントラップ内部のイオンをロッド間隙に向かわせるような電圧を印加する方法である。しかしながら、リニアイオントラップ外部の電極から電圧を印加したとしても、導体のロッドの存在により、イオンが蓄積されている領域まで十分に電場が到達せず、イオンを効率よく排出することは困難である。そこで図 2.11(c) に示すような、ロッド電極間に板状電極（挿入電極）を挿入し、この挿入電極にパルス電圧を印加してイオンを排出する方法が大阪大学で開発された [7]。この方法

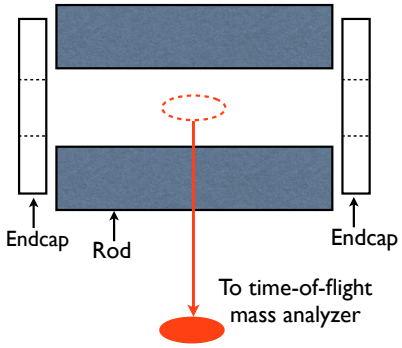
では、イオンが蓄積されている領域の近くまで挿入された挿入電極にイオン排出用のパルス電圧を印加する。したがって、イオンが蓄積されている領域に十分にイオン排出用の電場が到達するので、効率的にイオンを排出できる。イオンのトラップ時には、隣り合うロッド電極に 180 度異なる位相の高周波電圧が印加されているため、ロッド電極の中間では電位が一定である。そのため、挿入電極をロッド電極の中間位置に配置し、その電位をロッドの中間電位にしておけば、挿入電極が十分に薄ければ、イオンの蓄積に影響を与えない。また電源回路についても、挿入電極に RF 電圧を印加する必要がなく、印加する電圧はイオン排出用のパルス電圧のみであるため非常にシンプルである。



(a) リニアイオントラップからトラップ軸方向にイオンを排出する方法

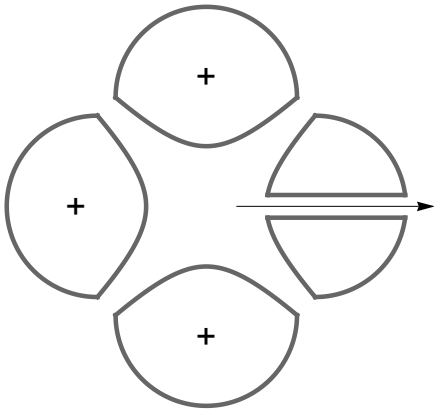


(b) リニアイオントラップからトラップ軸方向に排出したイオンを直交加速する方法

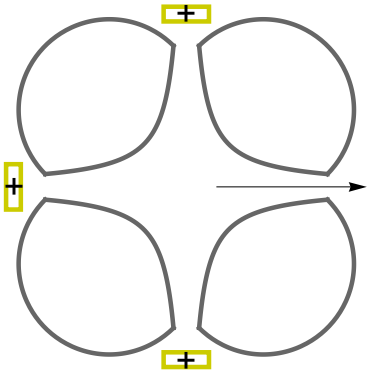


(c) リニアイオントラップからイオンをトラップ軸に直交する方向に排出する方法

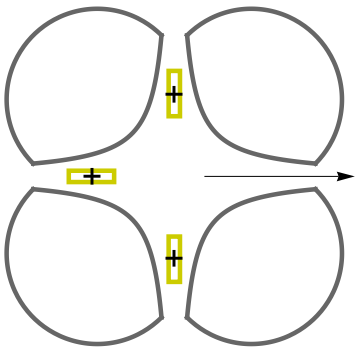
図 2.10: リニアイオントラップからのイオン排出方法



(a) ロッド電極に排出用の穴を空ける方法



(b) 外部電極に排出用電圧を印加する方法



(c) ロッド電極間に挿入した板状電極に排出用電圧
を印加する方法

図 2.11: リニアイオントラップからイオンをトラップ軸に直交する方向に排出する方法

ロッド電極間に板状電極を挿入したリニアイオントラップからトラップ内部に蓄積されたイオンを排出する際、検出器の位置での時間収束性を高めるために二段加速法を利用する [15]。二段加速法は、イオンの初期位置のばらつきによる飛行時間の差を収束させることができるイオンの加速法である。二段加速法は図 2.12 に示すように、イオンの加速を 2 つの段階に分けて、それぞれ異なる電場強度で加速する。電極 2 から任意の距離 s の位置にある、静止した質量 m、価数 Z のイオンが二段加速された場合の、検出器までの飛行時間 $T(s)$ は式 (2.13) で与えれる。

$$T(s) = \sqrt{\frac{m}{Ze}}\left\{\sqrt{\frac{2L_1 s}{V_1}} + L_3\sqrt{\frac{L_1}{2(sV_1 + L_1V_2)}} - \frac{L_2}{V_2}\sqrt{\frac{2sV_1}{L_1}} + \frac{L_2}{V_2}\sqrt{\frac{2(sV_1 + L_1V_2)}{L_1}}\right\} \tag{2.13}$$

ここで L_1, L_2, L_3 は図 2.12 に対応する各部の距離であり、V_1, V_2 は電極 1,2 への印加電圧を表す。さらに式 (2.13) を $s = L_1/2$ 周りでテーラー展開して、式 (2.14) を得る。

$$\begin{aligned} T(s) = T\Big(\frac{L_1}{2}\Big) + \sqrt{\frac{m}{ZeV_1}}\Bigg\{1 - \frac{L_2}{L1}\frac{V_1}{V_2} - \frac{L_3}{2\sqrt{2}L_1}\Big(\frac{1}{\frac{1}{2} + \frac{V_2}{V_1}}\Big)^{3/2} \\ + \frac{L_2}{\sqrt{2}L_1}\frac{V_1}{V_2}\sqrt{\frac{1}{\frac{1}{2} + \frac{V_2}{V_1}}}\Bigg\}\Big(s - \frac{L_1}{2}\Big) + \cdots \end{aligned} \tag{2.14}$$

式 (2.14) から、$T(s)$ の $s - \frac{L_1}{2}$ の 1 次の展開係数は $\frac{V_1}{V_2}$ を適当に調整すれば小さくなり、イオンの初期位置が電極 1,2 の真ん中からずれていても、飛行時間が 1 次のオーダーで収束することを意味している。

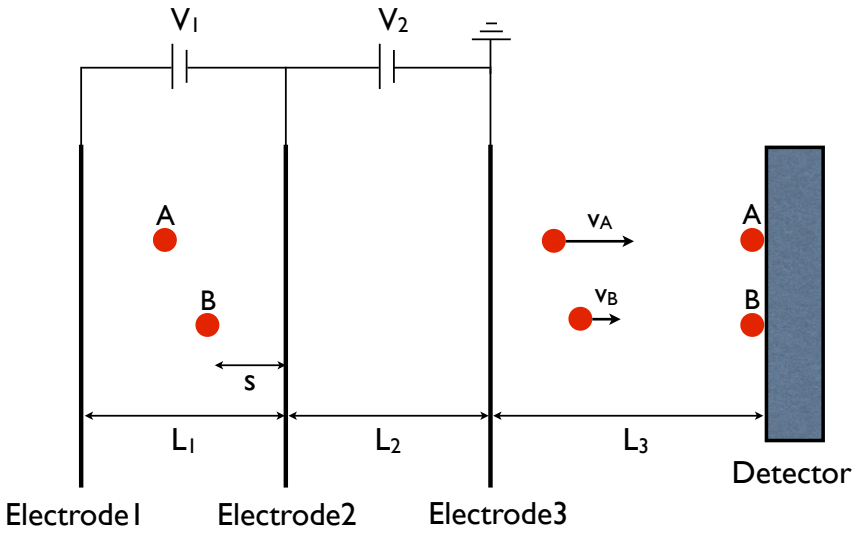


図 2.12: 2 段階加速法

開発した装置では図 2.13 示すように、ロッド電極間に板状電極を挿入したリニアイオントラップから排出されたイオンを引き出し電極 1,2 により二段加速できる構造になっており、イオンが排出されるときには図 2.14 に示したようなポテンシャルが形成される。しかしながら、イオンが二段加速され MULTUM-SⅡ に向けて飛行する際、引き出し電極 2 の後ろの電位は GND 電位であり、引き出し電極 2 に負の電圧が印加されていると減速されてしまう。そこで、引き出し電極 2 にはポテンシャルリフトの機能を持たせている [16]。ポテンシャルリフトは金属の筒でできており、筒の電位を変化させても筒を通過中のイオンの速度には影響を及ぼさないという原理を利用している。さらに、z 方向へのイオンの広がりを抑えるための、挿入電極 3 も設けている。各電極に印加する電圧については、図 2.15 にタイミングチャートを示す。ロッド電極にはイオンを蓄積する間は RF 電圧を印加し、その後 RF 電圧を停止し、瞬時に挿入電極および引き出し電極に高電圧を印加してイオンを排出する。

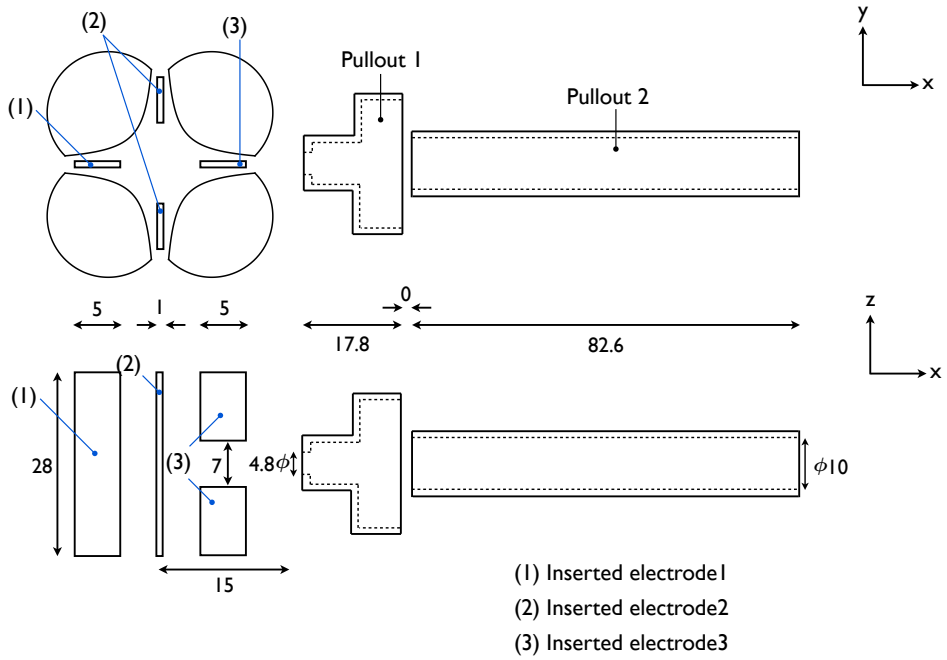


図 2.13: ロッド電極間に板状電極を挿入したリニアイオントラップと引き出し電極 1,2 の配置・形状 (エンドキャップ電極は省略)

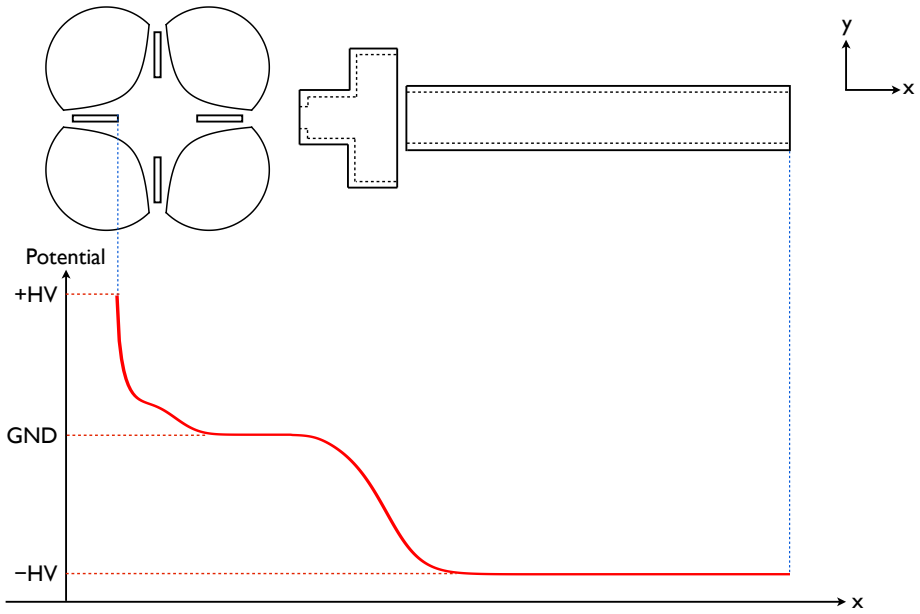


図 2.14: イオンを排出時の引き出し方向のポテンシャル分布

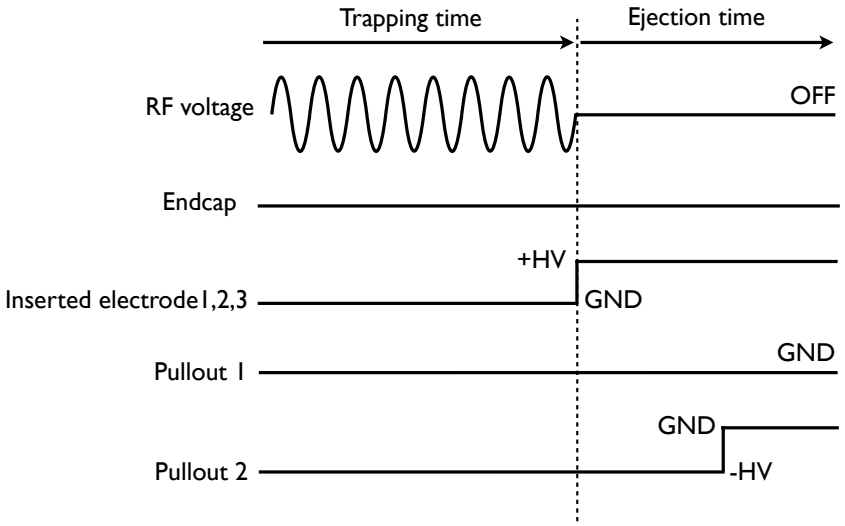


図 2.15: タイミングチャート

第3章　イオン軌道シミュレーション手法

ロッド電極間に板状電極を挿入したリニアイオントラップの性能を向上させるには、リニアイオントラップ内部でのイオンの空間分布・運動状態を正確に把握することが求められる。しかしながら、イオンの空間分布・運動状態を実験のみで見積もるのは現実的ではないため、数値シミュレーションによりイオンの空間分布・運動状態を見積もることにした。

まず、イオンは電極が形成する電場からクーロン力を受ける。そのため、イオンの軌道をシミュレーションする際に電極が形成する電場を計算しなければならない。また、イオンがリニアイオントラップに打ち込まれ図 2.9 に示したようなポテンシャルに閉じ込められるには、緩衝ガスと衝突してエネルギーを落とされている必要がある。そのため、イオンの軌道をシミュレーションする際に、イオンとガス分子との衝突を考慮する必要がある。そして、イオンの運動方程式を解いて、イオンの軌道・運動状態を計算する必要がある。

したがって、イオンの軌道をシミュレーションするのに必要な計算手法は次のものである。

・電極が形成する電場の計算

・衝突の計算

・電場中のイオンの運動方程式を解く

電極が形成する電場の計算手法にはいくつかの方法があるが、大きく領域分割法と境界分割法に分けられる。領域分割法には、差分法、有限要素法があり、境界分割法は電荷重畳法、表面電荷法がある。それぞれの電場計算法の特徴について簡単にまとめる。

差分法では、電極とその周囲の領域を平行な格子で分割する。つまり電極の形状を小さな四角形で模擬することになる。電極形状の模擬と、境界条件の方程式の作成の手順が非常にシンプルであるため、古くから使用されてきた方法である。しかしながら、四角形で電極の形状を模擬するために、複雑

な形状の電極を正確に表現することは困難である。

有限要素法では、領域を三角形で分割する。電極形状を三角形で模擬することになるため、複雑な形状の電極も正確に表現することができる。しかし、電極形状の模擬と、境界条件の方程式の作成の手順が複雑であることが難点である。また、領域分割法である差分法と有限要素法の共通の問題として、電位の計算誤差に比べて電場の計算誤差がかなり大きくなることがあげられる。これは領域分割法が、電位を数値微分して電場を計算することにより生じる問題である。

一方で、境界要素法である電荷重畳法と表面電荷法は、電位を数値微分して電場を計算するというプロセスは不要であり、領域分割法に比べ電場の計算誤差が小さいという長所がある。また、境界要素法では、電極形状の模擬が電極表面のみを分割すれば事足りるため、電極形状の模擬のプロセスが領域分割法に比べて圧倒的に簡易である。複雑な形状の電極の寸法を容易に変更してシミュレーションできることは、非常に大きなメリットである。

境界分割法の候補として電荷重畳法と表面電荷法を挙げたが、本研究では表面電荷法を使用した。これは表面電荷法は、電荷重畳法では計算が難しい形状、例えば、極端に薄い形状の電極なども容易に計算できるからである [10]。本章では電場の計算法に関しては、まず表面電荷法の基本的な考え方を説明し、２次元場の表面電荷法と、さらに一般の３次元場の表面電荷法について説明する。

イオンと中性のガス粒子との衝突の取り扱いについては、確率的な事象であるため、平均自由行程の概念を用いたモンテカルロ法を適用できるモデルについて説明する。

イオンの運動方程式を数値的に積分する手法として、４次のルンゲ・クッタ法がよく用いられる。本章では、４次のルンゲ・クッタ法をイオンの運動方程式に適用する方法を説明する。

3.1 表面電荷法の基本的な考え方

静電場中では電極の電荷はすべて表面に存在する。表面の微小な面積 ΔS における、表面電荷密度を σ とすると、この微小面積が空間に形成する電位は

$$\phi = \frac{\sigma \Delta S}{4\pi\varepsilon_0 l} \tag{3.1}$$

と表される。ここで l は空間の任意の点と、微小面積までの距離である (図 3.1)。

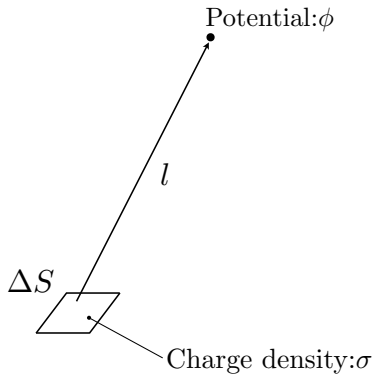


図 3.1: 表面電荷密度が σ の微小面積 ΔS が空間に形成する電位

次に電極の表面全体を、表面電荷密度が一定の値 σ_j と見なせる微小面積 (要素)ΔS_j に分割した状況を考える (図 3.2)。電極全体が形成する電位を計算するには、すべての要素からの作用を足し上げればよい。したがって、空間のある点 i の電位は

$$\phi_i = \frac{1}{4\pi\varepsilon_0} \sum_j \frac{\sigma_j \Delta S_j}{l_{ij}} \tag{3.2}$$

と表される。l_{ij} は点 i と各要素の距離を表す。このようにして、各要素の電荷密度 σ_j と i 点の電位 ϕ_i との関係式が得られる。

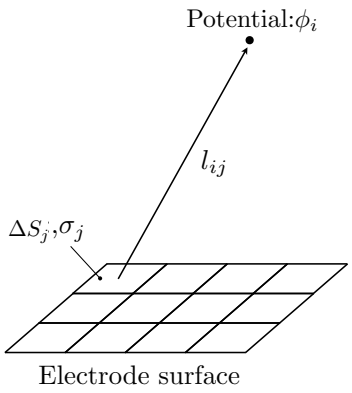


図 3.2: 電極全体が形成する電位を分割要素の寄与を足し上げて計算

ここで i 点を電極表面に配置し、電極の電位が V_i であるとすれば

$$V_i = \frac{1}{4\pi\varepsilon_0} \sum_j \frac{\sigma_j \Delta S_j}{l_{ij}} \tag{3.3}$$

となる。さらに電極表面に要素と同じ数だけ V_i の点を配置すれば

$$\begin{bmatrix} \frac{\Delta S_1}{l_{1,1}} & \cdots & \frac{\Delta S_n}{l_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\Delta S_1}{l_{n,1}} & \cdots & \frac{\Delta S_n}{l_{n,n}} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix} = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \tag{3.4}$$

という電極表面の境界条件を表す連立一次方程式が得られる。ここで左辺の行列

$$\begin{bmatrix} \frac{\Delta S_1}{l_{1,1}} & \cdots & \frac{\Delta S_n}{l_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\Delta S_1}{l_{n,1}} & \cdots & \frac{\Delta S_n}{l_{n,n}} \end{bmatrix}$$

は電極を分割する要素の形状と、要素と電圧を与える i 点との位置関係のみで決まり、行列のそれぞれの成分は「電位係数」と呼ばれる。例えばこの行列の i 行 j 列の成分は、j 番の要素が i 点に対して形成する電位係数である、などと表現する。さらに電極表面の電位 V_1〜V_n を与えれば、式 (3.61) を解くことができ、表面電荷密度 σ_1〜σ_n の値が計算できる。電極表面の各要素の表面電荷密度を求めたので、それらの作用を全て足し上げることで、任意の点に電極が形成する電位・電場を計算できる。

以上が表面電荷法の基本的な考え方である。ただし、表面電荷密度は各要素内で一定とはせず、座標の関数として変化させる方が、隣り合う要素同士

で表面電荷密度が連続になるため、電場の計算精度が向上する。このとき、表面電荷密度の作用は式 (3.1) のように簡単ではなく、各要素で積分した電位としなくてはならない。

3.2　３次元場の計算

3.2.1　３次元場の表面電荷法

一般３次元形状の電極が形成する電場の計算法について説明する。電極表面の形状は三角形の要素 (element) によって模擬する。三角形の要素を用いれば、複雑な形状の電極もよく模擬することができる。また三角形要素の頂点は節点 (node) と呼称する (図 3.3)。

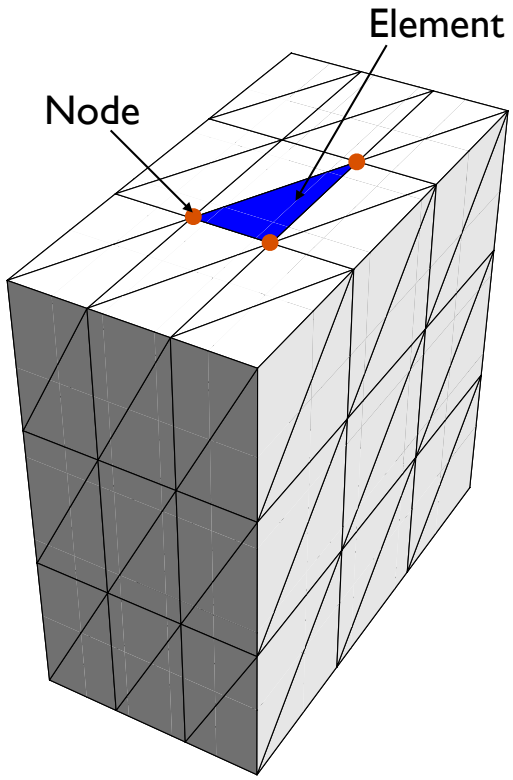


図 3.3: 三角形要素による電極形状の模擬の例

各々の三角形要素に番号付けを行い、e 番目の要素を E_e $(e = 1, \cdots, m)$ と表す。節点も番号付けを行い、i 番目の節点を P_i $(i = 1, \cdots, n)$ と表す。また i 番目の節点の位置ベクトルを $\boldsymbol{R}_i$、表面電荷密度を σ_i と表す。

三角形要素内部の電荷密度は、三角形内部の座標の１次式として与える。このように表面電荷密度を定めることで、隣合う要素同士で表面電荷密度がなめらかに接続され、要素内部で表面電荷密度を一定とする場合よりも、電

位・電場の計算精度が向上する。まず、三角形要素上の表面電荷密度を座標の 1 次式として与える数学的な表現を導く。

節点 P_i, P_j, P_k を含む要素 E_e の電荷密度 $\sigma_e(\boldsymbol{R})$ を式 (3.5) で表現する。

$$\sigma_e(\boldsymbol{R}) = f_{ie}(\boldsymbol{R})\sigma_i + f_{je}(\boldsymbol{R})\sigma_j + f_{ke}(\boldsymbol{R})\sigma_k \tag{3.5}$$

$\boldsymbol{R}$ は要素 E_e 上の任意の座標である。要素内部の表面電荷密度を座標 $\boldsymbol{R}$ の 1 次式として与えたいので、関数 $f_{ie}(\boldsymbol{R}), f_{je}(\boldsymbol{R}), f_{ke}(\boldsymbol{R})$ を座標 $\boldsymbol{R}$ の 1 次式となるように定めなければならない (図 3.4)。

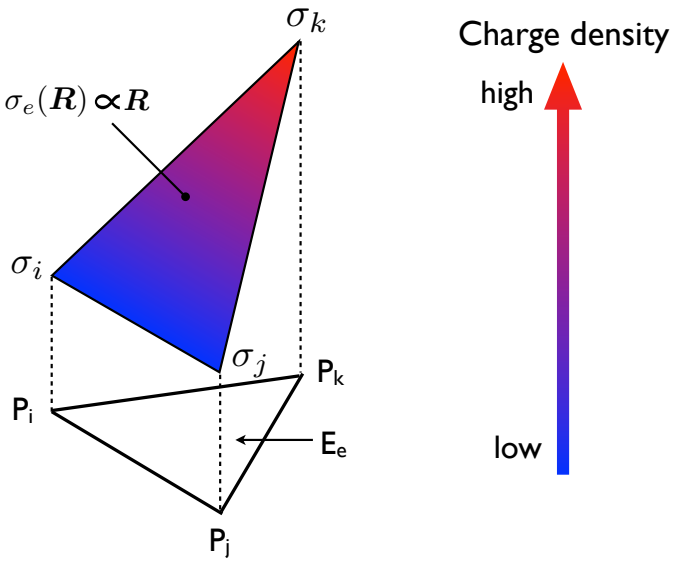


図 3.4: 三角形要素内部での表面電荷密度の分布

要素 E_e 上の節点 P_i, P_j, P_k の表面電荷密度 $\sigma_i, \sigma_j, \sigma_k$ は、$f_{ie}(\boldsymbol{R}), f_{je}(\boldsymbol{R}), f_{ke}(\boldsymbol{R})$ 用いると式 (3.24) のように表される。

$$\begin{aligned}
\sigma_i &= \sigma_e(\boldsymbol{R_i}) = f_{ie}(\boldsymbol{R_i})\sigma_i + f_{je}(\boldsymbol{R_i})\sigma_j + f_{ke}(\boldsymbol{R_i})\sigma_k \\
\sigma_j &= \sigma_e(\boldsymbol{R_j}) = f_{ie}(\boldsymbol{R_j})\sigma_i + f_{je}(\boldsymbol{R_j})\sigma_j + f_{ke}(\boldsymbol{R_j})\sigma_k \\
\sigma_k &= \sigma_e(\boldsymbol{R_k}) = f_{ie}(\boldsymbol{R_k})\sigma_i + f_{je}(\boldsymbol{R_k})\sigma_j + f_{ke}(\boldsymbol{R_k})\sigma_k
\end{aligned} \tag{3.6}$$

このとき、$f_{ie}(\boldsymbol{R}), f_{je}(\boldsymbol{R}), f_{ke}(\boldsymbol{R})$ は式 (3.61) を満たしていなければならない。

$$\begin{aligned}
&f_{ie}(\boldsymbol{R_i}) = 1 \quad && f_{je}(\boldsymbol{R_i}) = 0 \quad && f_{ke}(\boldsymbol{R_i}) = 0 \\
&f_{ie}(\boldsymbol{R_j}) = 0 \quad && f_{je}(\boldsymbol{R_j}) = 1 \quad && f_{ke}(\boldsymbol{R_j}) = 0 \\
&f_{ie}(\boldsymbol{R_k}) = 0 \quad && f_{je}(\boldsymbol{R_k}) = 0 \quad && f_{ke}(\boldsymbol{R_k}) = 1
\end{aligned} \tag{3.7}$$

ここで $\boldsymbol{R}$ は、媒介変数 u_1, u_2 を導入して式 (3.8) のように表現することができる。

$$\boldsymbol{R} = \boldsymbol{R_i} + \frac{1}{2}(1+u_1)(\boldsymbol{R_j} - \boldsymbol{R_i}) + \frac{1}{4}(1+u_1)(1+u_2)(\boldsymbol{R_k} - \boldsymbol{R_j}) \qquad (3.8)$$
$$(-1 < u_1 < 1, -1 < u_2 < 1)$$

$\boldsymbol{R}$ を式 (3.8) で定めた場合、$f_{ie}(\boldsymbol{R}), f_{je}(\boldsymbol{R}), f_{ke}(\boldsymbol{R})$ は式 (3.9)〜(3.11) に決まる。

$$f_{ie}(\boldsymbol{R}) = \frac{(1-u_1)}{2} \qquad (3.9)$$

$$f_{je}(\boldsymbol{R}) = \frac{(1+u_1)(1-u_2)}{4} \qquad (3.10)$$

$$f_{ke}(\boldsymbol{R}) = \frac{(1+u_1)(1+u_2)}{4} \qquad (3.11)$$

式 (3.9)〜(3.11) は、式 (3.61) の条件を満たしており、図 3.5 に示すように $\boldsymbol{R}$ に比例して変化するので適切である。

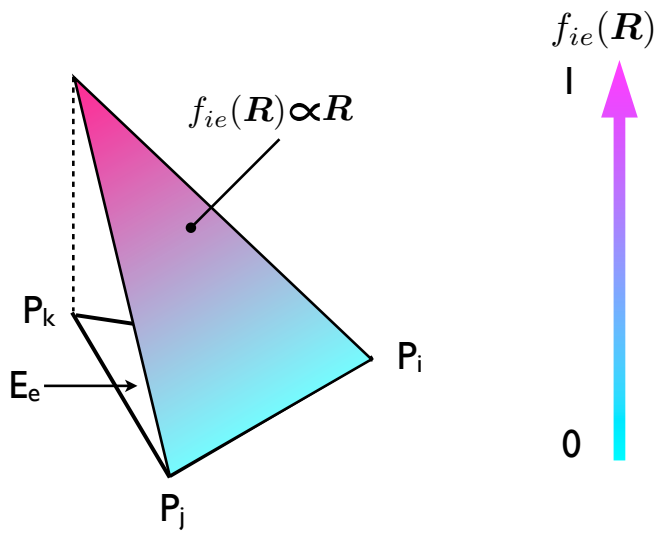


図 3.5: 三角形要素内部での関数 $f_{ie}(\boldsymbol{R})$ の値

したがって、電極全体の表面電荷密度の分布 $\sigma(\boldsymbol{R})$ は式 (3.12) で表すことができる。

$$\sigma(\boldsymbol{R}) = \sum_{e=1}^{m} \sigma_e(\boldsymbol{R}) = \sum_{i=1}^{n} \sigma_i \sum_{e=1}^{m} f_{ie}(\boldsymbol{R}) \qquad (3.12)$$

ただし要素 E_e が節点 P_i を含まない場合、$f_{ie}(\boldsymbol{R}) = 0$ とする。

式 (3.12) から電極全体の表面電荷密度分布は、節点上での表面電荷密度が与えられれば、一意に決まることが分かる。そこで、節点上の表面電荷密度を未知数とした、電極表面の境界条件を表す連立一次方程式を導く。節点 P_i のポテンシャルが V_i と与えられたとすれば式 (3.13) が成り立つ。

$$\begin{aligned} V_i &= V(\boldsymbol{R}_i) \\ &= \int \frac{\sigma(\boldsymbol{R})}{|\boldsymbol{R_i} - \boldsymbol{R}|} ds \\ &= \sum_{i=1}^{n} \sigma_i \sum_{e=1}^{m} \int_{S_e} \frac{f_{ie}(\boldsymbol{R})}{|\boldsymbol{R_i} - \boldsymbol{R}|} ds \\ &= \sum_{i=1}^{n} \sigma_i \sum_{e=1}^{m} A_{ije} \\ &= \sum_{j=1}^{n} A_{ij} \sigma_i \end{aligned} \tag{3.13}$$

簡単のために真空の誘電率 ε_0 を $\varepsilon_0 = \frac{1}{4\pi}$ としている。A_{ije},A_{ij} はそれぞれ式 (3.14),(3.15) で表される。

$$A_{ije} = \int_{S_e} \frac{f_{ie}(\boldsymbol{R})}{|\boldsymbol{R_i} - \boldsymbol{R}|} ds \tag{3.14}$$

$$A_{ij} = \sum_{e=1}^{m} A_{ije} \tag{3.15}$$

A_{ije} は節点 P_j を含む要素のひとつである E_e の表面電荷が、節点 P_i に形成する電位係数である。A_{ij} は節点 P_j を含む全ての要素の表面電荷が節点 P_i に形成する電位係数なので、A_{ije} を足し合わせたものとなる。

A_{ije} を解析的に計算することは困難なため、数値積分を用いて計算する。まず三角形要素上の面積分を、媒介変数 $-1 < u_1 < 1, -1 < u_2 < 1$ を用いた積分に変換し式 (3.16) を得る。

$$\begin{aligned} A_{ije} &= \int_{S_e} \frac{f_{ie}(\boldsymbol{R})}{|\boldsymbol{R_i} - \boldsymbol{R}|} ds \\ &= \frac{S_e}{4} \int_{-1}^{1} \int_{-1}^{1} (1 + u_1) \frac{f_{ie}(\boldsymbol{R}(u_1, u_2))}{|\boldsymbol{R_i} - \boldsymbol{R}(u_1, u_2)|} du_1 du_2 \end{aligned} \tag{3.16}$$

S_e は要素 E_e の面積である。式 (3.16) にはガウスの数値積分公式 [17] を適用することができて式 (3.17) を得る。

$$A_{ije} = \frac{S_e}{4} \sum_{a=1}^{N} \sum_{b=1}^{N} w_a w_b (1 + u_{1a}) \frac{f_{ie}(\boldsymbol{R}(u_{1a}, u_{2b}))}{|\boldsymbol{R_i} - \boldsymbol{R}(u_{1a}, u_{2b})|} \tag{3.17}$$

ここで式 (3.8) から $\boldsymbol{R}(u_{1a}, u_{2b})$ は式 (3.18) で表される。

$$\boldsymbol{R}(u_{1a}, u_{2b}) = \boldsymbol{R}_i + \frac{1}{2}(1+u_{1a})(\boldsymbol{R}_j - \boldsymbol{R}_i) + \frac{1}{4}(1+u_{1a})(1+u_{2b})(\boldsymbol{R}_k - \boldsymbol{R}_j) \tag{3.18}$$

N はガウスの数値積分の積分点数、u_{1a}, u_{2b} はガウスの数値積分の積分点、w_a, w_b は重みを表す。$N = 3$ のガウスの数値積分公式を用いたときの $\boldsymbol{R}(u_{1a}, u_{2b})$ を図示すると図 3.6 のようになる。

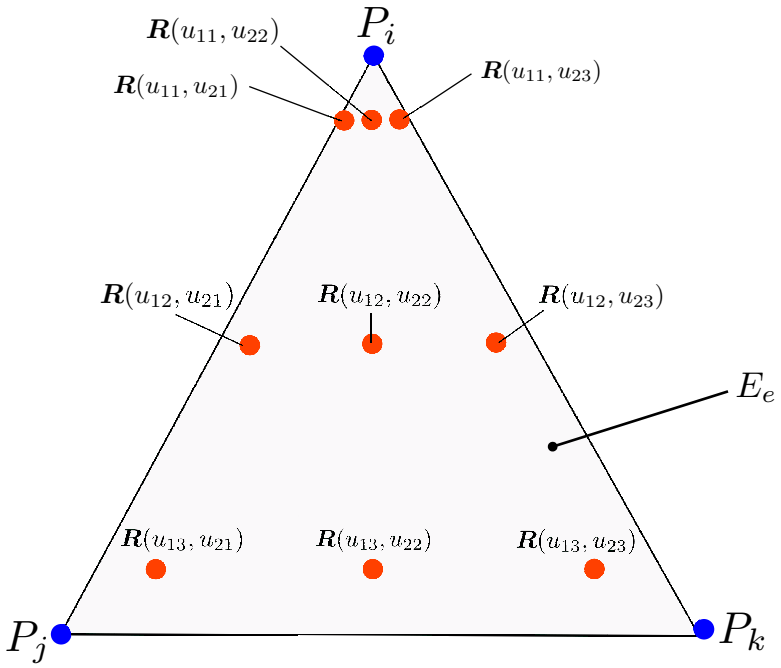


図 3.6: 3 点のガウスの数値積分公式を用いたときの $\boldsymbol{R}(u_{1a}, u_{2b})$ の分布

A_{ije} を数値積分によってもとめることは、図 3.6 に示した $\boldsymbol{R}(u_{1a}, u_{2b})$ の位置に点電荷 (仮想電荷) を配置し、仮想電荷によって E_e の表面電荷の作用を近似していることと同じである。それぞれの仮想電荷の電荷量は式 (3.19) によって与えられる。

$$\frac{S_e}{4} w_a w_b (1+u_{1a}) \{\sigma_i f_{ie}(u_{1a}, u_{2b}) + \sigma_j f_{je}(u_{1a}, u_{2b}) + \sigma_k f_{ke}(u_{1a}, u_{2b})\} \tag{3.19}$$

式 (3.17) より電位係数 A_{ij} を計算し、節点の表面電荷密度を未知数とした、

電極表面での境界条件を表す連立一次方程式 (3.20) をつくることができる。

$$\begin{bmatrix} A_{11} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix} = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \tag{3.20}$$

連立一次方程式 (3.20) を解き、電極表面の全ての節点の表面電荷密度 σ_1〜σ_n を求めることができる。

表面電荷密度 σ_1〜σ_n が計算できたので、式 (3.19) から全ての要素表面に配置した仮想電荷の電荷量を計算することができる。表面電荷の作用を仮想電荷によって近似しているので、任意の位置に電極が形成する電場は、全ての仮想電荷の寄与をたし上げることで計算できる (図 3.7)。仮想電荷は点電荷であったので電場は式 (3.21) で与えられる。

$$\boldsymbol{E}(\boldsymbol{r}) = \sum_{t=1}^{mN^2} q_t \frac{\boldsymbol{r} - \boldsymbol{r}_t}{|\boldsymbol{r} - \boldsymbol{r}_t|^3} \tag{3.21}$$

ここで q_t は仮想電荷の電荷量、$\boldsymbol{r}_t$ は仮想電荷の座標を表す。

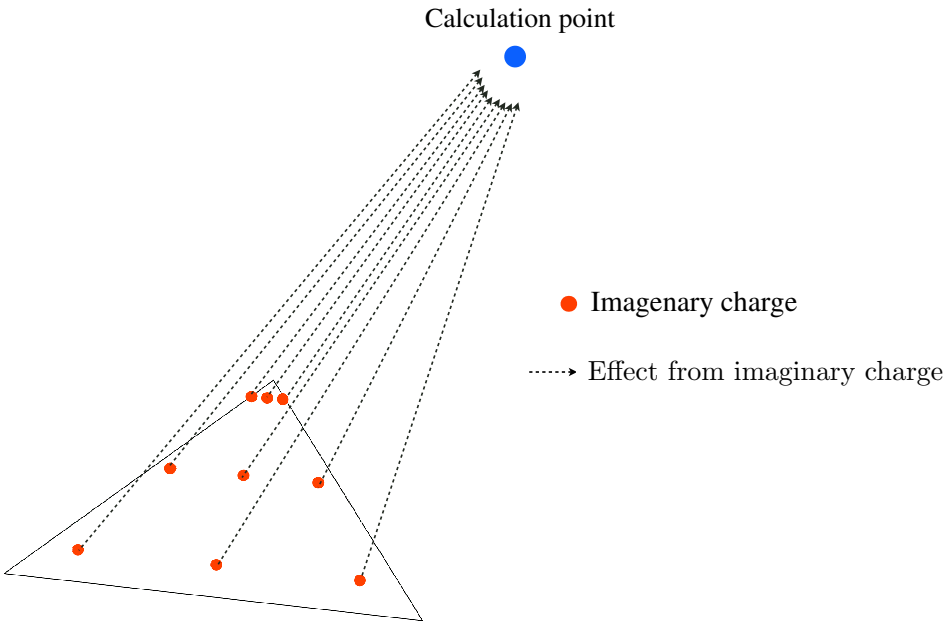


図 3.7: 仮想電荷の作用を足し上げることによる電場の計算

3.2.2　３次元場の表面電荷法プログラムの確認

C++にて３次元場の表面電荷法プログラムを作成し、解析解が与えられる電極形状について、表面電荷法で計算した電場を解析解と比較し、プログラムの動作チェックを行った。

同軸の二重の円筒電極が形成する電場の解析解は式 (3.22) で与えられる。

$$E(r) = -\frac{1}{r}\frac{V_{in} - V_{out}}{log(r_{in}) - log(r_{out})} \tag{3.22}$$

ここで r は動径方向の位置、r_{in}, r_{out} は内側, 外側の電極の半径、V_{in}, V_{out} は内側, 外側の電極の電圧である。図 3.8 に示すような、$r_{in} = 1.0$[mm],$r_{out} = 2.0$[mm], 高さ 10[mm] の二重円筒に、$V_{in} = -1.0$[V],$V_{out} = 1.0$[V] の電圧を印加したときの動径方向の電場を計算した。表面電荷法で電場を計算した電場と解析解を図 3.9 に示す。また表面電荷法により計算した電場の解析解との誤差を図 3.10 に示す。解析解と表面電荷法で計算した電場が 1 %以下で一致しており、３次元場の表面電荷法プログラムが正常に動作していることを確認した。

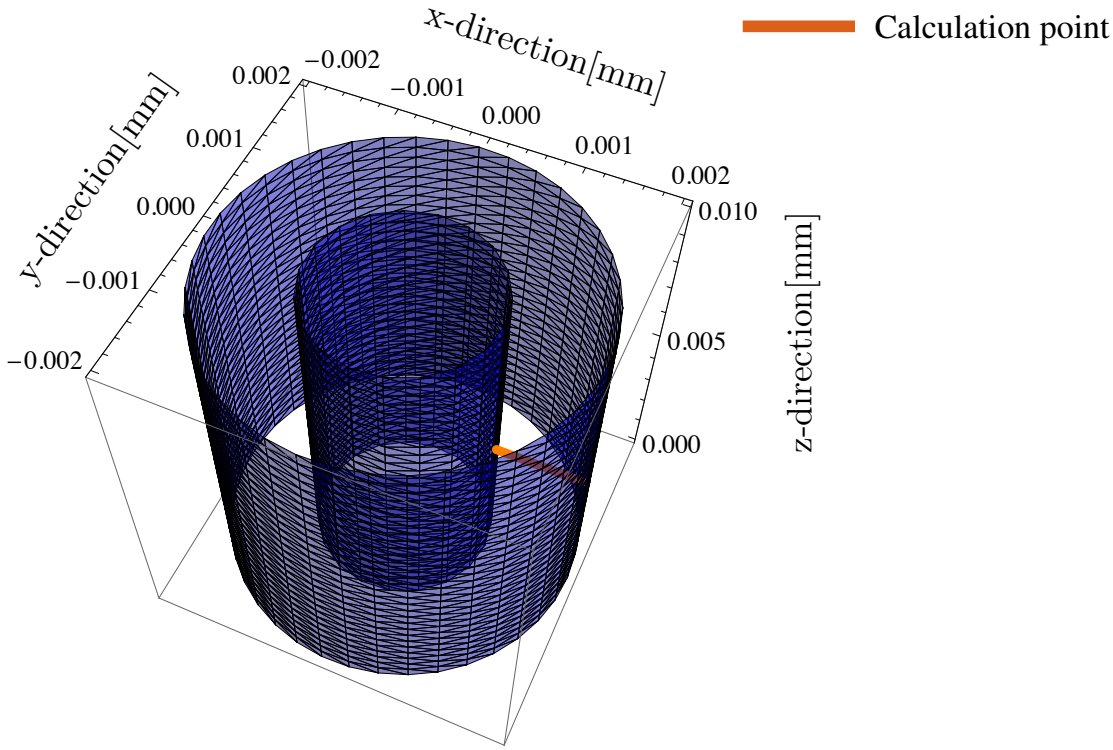


図 3.8: 二重円筒の模擬と電場の計算点の配置

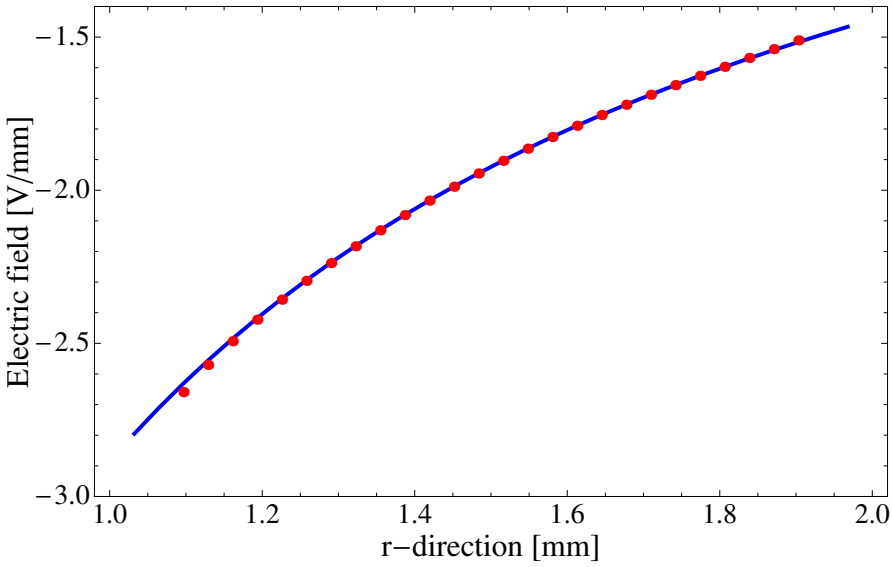


図 3.9: 表面電荷法の計算結果と解析解の比較

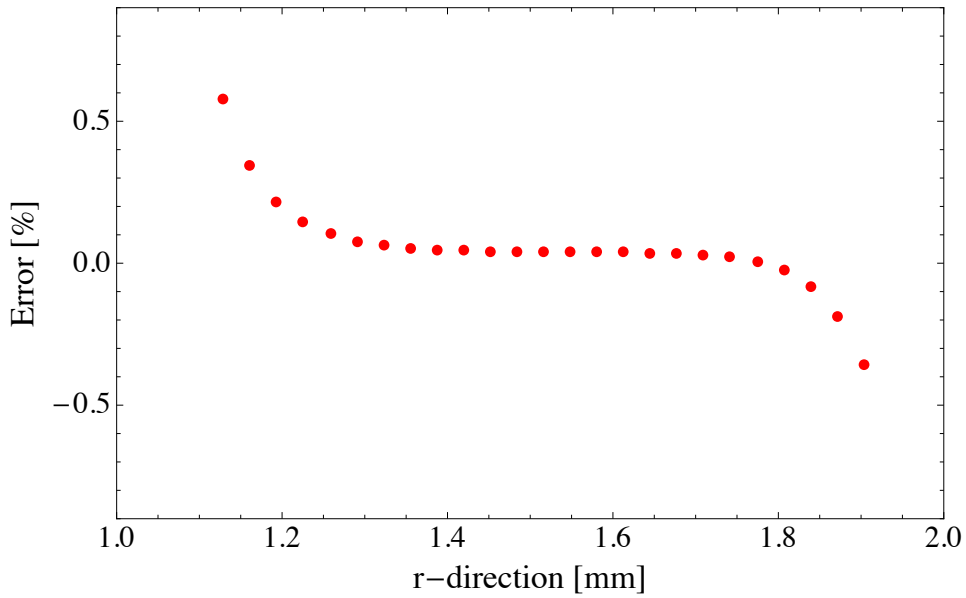


図 3.10: 表面電荷法の誤差

3.3　２次元場の計算

3.3.1　２次元場の表面電荷法

２次元場とはある平面に垂直で、無限に長い電極が形成する電場を意味する。電極が十分に長く端電場の影響が無視できるようなときは、３次元場の表面電荷法を使用するより、２次元場の表面電荷法を用いた方がはるかに高速に電場を計算できるため有用である。ここでは２次元場の表面電荷法について説明する。

電極表面の形状は無限に長い帯状の要素によって模擬し、帯状の要素の端は節点と呼ぶことにする (図 3.11)。

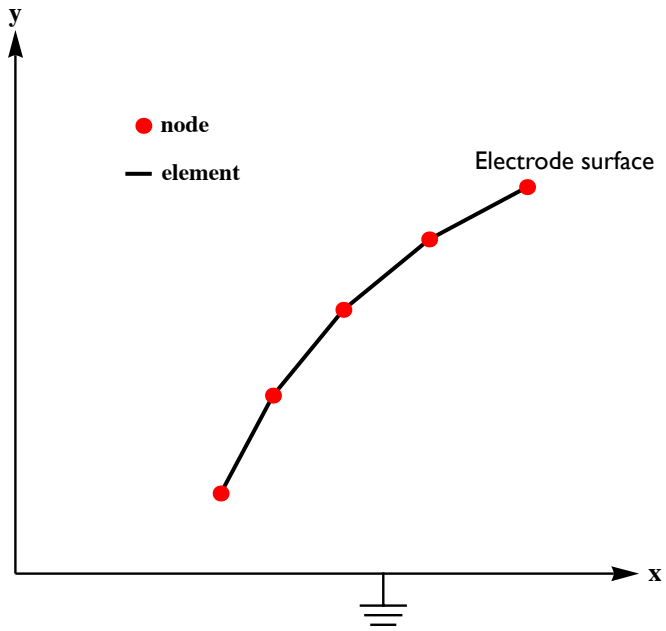


図 3.11: 無限に長い帯状要素による電極模擬の例 (断面図)

各々の要素に番号付けを行い、e 番目の要素を E_e $(e = 1, \cdots, m)$ と表す。節点も番号付けを行い、i 番目の節点を P_i $(i = 1, \cdots, n)$ と表し、その位置ベクトルを (X_i, Y_i)、表面電荷密度を σ_i と表す。

要素内部の表面電荷密度は、座標 (図 3.11 で言えば x, y) の１次式として与える。このように定めることで、隣あう要素同士で表面電荷密度がなめらかに接続され、ひとつの要素内部で表面電荷密度が一定であるとする場合より

も、電位・電場の計算精度が向上する。まず、要素上の表面電荷密度を座標の１次式として与える数学的な表現を導く。

節点 P_i, P_j を含む要素 E_e の表面電荷密度 $\sigma_e(x, y)$ を式 (3.23) のように表す。

$$\sigma_e(x, y) = f_{ie}(x, y)\sigma_i + f_{je}(x, y)\sigma_j \tag{3.23}$$

x, y は要素上の任意の座標である。要素内部の表面電荷密度を座標 (x, y) の１次式で与えたいので、f_{ie},f_{je} も (x, y) の１次式でとなるように定めなければならない (図 3.12)。

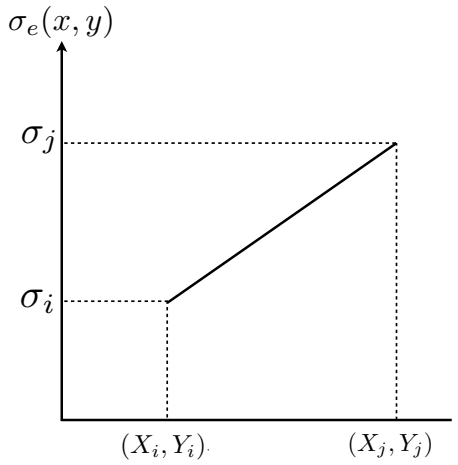


図 3.12: 節点 P_i, P_j を含む要素 E_e 上での表面電荷密度の分布

要素 E_e 上の節点 P_i, P_j の表面電荷密度 σ_i, σ_j は、f_{ie},f_{je} を用いて式 (3.24) のように表される。

$$\begin{aligned} \sigma_i &= \sigma_e(X_i, Y_i) = f_{ie}(X_i, Y_i)\sigma_i + f_{je}(X_i, Y_i)\sigma_j \\ \sigma_j &= \sigma_e(X_j, Y_j) = f_{ie}(X_j, Y_j)\sigma_i + f_{je}(X_j, Y_j)\sigma_j \end{aligned} \tag{3.24}$$

このとき、f_{ie},f_{je} は式 (3.25) を満たしていなければならない。

$$\begin{aligned} f_{ie}(X_i, Y_i) &= 1 \qquad f_{je}(X_i, Y_i) = 0 \\ f_{ie}(X_j, Y_j) &= 0 \qquad f_{je}(X_j, Y_j) = 1 \end{aligned} \tag{3.25}$$

ここで要素 E_e 上の任意の座標 (x, y) は、媒介変数 u $(-1 < u < 1)$ を導入し

て式 (3.26) のように表現できる。

$$\begin{aligned} x(u) &= \frac{1-u}{2}X_i + \frac{1+u}{2}X_j \\ y(u) &= \frac{1-u}{2}Y_i + \frac{1+u}{2}Y_j \end{aligned} \tag{3.26}$$

(x, y) を式 (3.26) で定めた場合、$f_{ie}(x, y), f_{je}(x, y)$ は式 (3.27),(3.28) に決まる。

$$f_{ie}\{x(u), y(u)\} = \frac{1-u}{2} \tag{3.27}$$

$$f_{je}\{x(u), y(u)\} = \frac{1+u}{2} \tag{3.28}$$

式 (3.27),(3.28) は、式 (3.25) の条件を満たしており、図 3.13 に示すように u に比例して変化するので適切である。

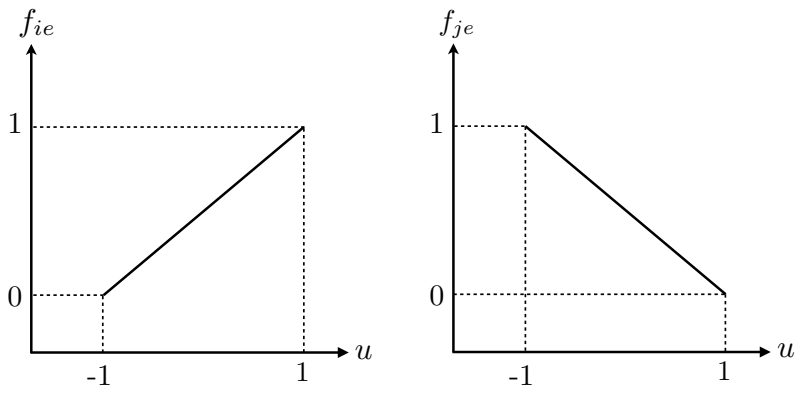


図 3.13: 要素上 $(-1 < u < 1)$ での関数 f_{ie}, f_{je}の値

したがって、電極全体での表面電荷密度の分布 $\sigma(x, y)$ は、式 (3.12) と同様にして式 (3.29) で表すことができる。

$$\sigma(x, y) = \sum_{i=1}^{n} \sigma_i \sum_{e=1}^{m} f_{ie}(x, y) \tag{3.29}$$

ただし、要素 E_e が節点 P_i を含まない場合、$f_{ie}(x, y) = 0$ とする。式 (3.29) から電極全体の表面電荷密度分布は、節点の表面電荷密度が与えられれば、一意に決まることが分かる。そこで、節点の表面電荷密度を未知数とした、電極表面の境界条件を表す連立一次方程式を導く。

(x, y) 平面に垂直で無限長の、線電荷密度 λ の線電荷が (X, Y) にある場合のポテンシャル分布は式 (3.30) で表される。

$$
\begin{aligned}
V(x,y) &= \frac{\lambda}{2}\Big[log\big\{(x-X)^2+(y-Y)^2\big\} - log\big\{(x-X)^2+(y+Y)^2\big\}\Big] \\
&= \lambda F(x,y,X,Y) \qquad (3.30)
\end{aligned}
$$

簡単のために真空の誘電率 ε_0 を $\varepsilon_0 = \frac{1}{2\pi}$ としている。関数 $F(x,y,X,Y)$ は式 (3.31) で定義する。

$$
F(x,y,X,Y) = \frac{1}{2}\Big[log\big\{(x-X)^2+(y-Y)^2\big\} - log\big\{(x-X)^2+(y+Y)^2\big\}\Big] \qquad (3.31)
$$

この表現は $(X,-Y)$ にある鏡像電荷の影響も含むので x 軸がアース電位となり、式 (3.32) の境界条件を満たす。

$$
V(x,y) = 0 \quad (\ \sqrt{(x-X)^2+(y-Y)^2} \ \rightarrow \ \infty \ ) \qquad (3.32)
$$

表面電荷密度分布が与えられれば式 (3.30) を用いて、式 (3.33) のようにポテンシャルを計算できる。

$$
V(x,y) = \int F(x,y,X,Y)d\lambda = \int F(x,y,X,Y)\sigma dl \qquad (3.33)
$$

したがって、節点 P_i のポテンシャルが V_i と与えれば式 (3.34) が成り立つ。

$$
\begin{aligned}
V_i &= V(X_i,Y_i) \\
&= \sum_{j=1}^{n}\sigma_j \sum_{e=1}^{m}\int_{l_e} f_{je}(X,Y)F(X_i,Y_i,X,Y)dl \\
&= \sum_{j=1}^{n}\sigma_j \sum_{e=1}^{m} A_{ije} \\
&= \sum_{j=1}^{n} A_{ij}\sigma_j \qquad (3.34)
\end{aligned}
$$

A_{ije}, A_{ij} はそれぞれ式 (3.37),(3.36) で表される。

$$
A_{ije} = \int_{l_e} f_{je}(X,Y)F(X_i,Y_i,X,Y)dl \qquad (3.35)
$$

$$
A_{ij} = \sum_{e=1}^{m} A_{ije} \qquad (3.36)
$$

3 次元の場合と同様に、A_{ije} は節点 P_j を含む要素のひとつである E_e の表面電荷が、節点 P_i に形成する電位係数である。A_{ij} は節点 P_j を含む全ての要素の表面電荷が節点 P_i に形成する電位係数なので、A_{ije} を足し合わせたものである。

A_{ije} は式 (3.37) を媒介変数 u の積分に変換し、その後ガウスの数値積分公式を適用して計算する。

$$\begin{aligned} A_{ije} &= \int_{l_e} f_{je}(X,Y)F(X_i,Y_i,X,Y)dl \\ &= l_e \int_{-1}^{1} f_{je}(X(u),Y(u))F(X_i,Y_i,X(u),Y(u))du \\ &= l_e \sum_{k=1}^{N} w_k f_{je}\{X(u_k),Y(u_k)\}F\{X_i,Y_i,X(u_k),Y(u_k)\} \end{aligned} \quad (3.37)$$

N はガウスの数値積分の積分点数、u_k は積分点、w_k は重みを表す。l_e は式 (3.74) で与えられる。

$$l_e = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \quad (3.38)$$

ここで式 (3.27),(3.28) から $(X(u_k), Y(u_k))$ は、式 (3.39),(3.40) で表される。

$$X(u_k) = \frac{1-u_k}{2}X_i + \frac{1+u_k}{2}X_j \quad (3.39)$$

$$Y(u_k) = \frac{1-u_k}{2}Y_i + \frac{1+u_k}{2}Y_j \quad (3.40)$$

$N = 3$ のガウスの数値積分公式を用いたときの $(X(u_k), Y(u_k))$ を図示すると図 3.14 のようになる。

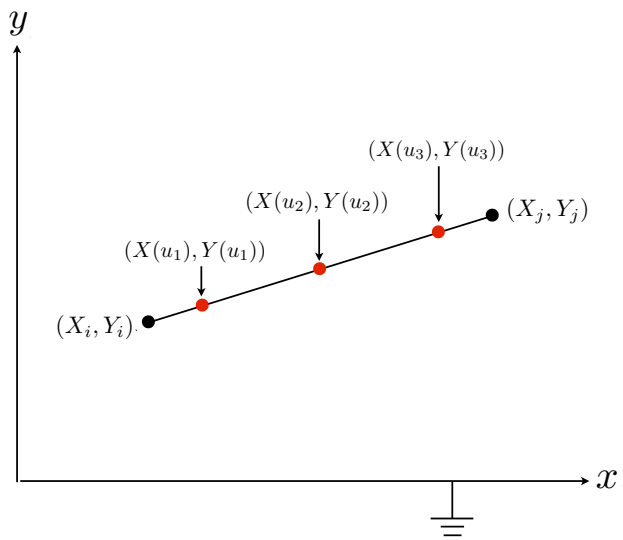


図 3.14: 3 点のガウスの数値積分公式を用いたときの $(X(u_k), Y(u_k))$ の位置

A_{ije} を数値積分で求めることは、図 3.14 に示した $(X(u_k), Y(u_k))$ の位置に、無限に長い線電荷（仮想電荷）を配置し、仮想電荷によって E_e の表面電荷

の作用を近似していることと同じである。それぞれの仮想電荷の線電荷密度 λ_k は式 (3.41) によって与えられる。

$$\lambda_k = l_e \sigma_i f_{ie}\{X(u_k), Y(u_k)\} + l_e \sigma_j f_{je}\{X(u_k), Y(u_k)\} \tag{3.41}$$

式 (3.36),(3.37) を用い、節点の表面電荷密度を未知数とした電極表面の境界条件を表す連立方程式 (3.42) をつくることができる。

$$\begin{bmatrix} A_{11} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix} = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \tag{3.42}$$

連立方程式 (3.42) を解き、表面電荷密度 σ_1〜σ_n をもとめることができる。

表面電荷密度 σ_1〜σ_n が計算できたので、式 (3.41) から全ての要素表面に配置した仮想電荷の電荷量を計算することができる。。表面電荷の作用を仮想電荷によって近似しているので、任意の位置に電極が形成する電場は、全ての仮想電荷の寄与をたし上げることで計算でき、式 (3.43),(3.44) のようになる。

$$E_x(x,y) = \sum_{t=1}^{mN} \lambda(t)(x - X_t)\left(\frac{1}{L_-} - \frac{1}{L_+}\right) \tag{3.43}$$

$$E_y(x,y) = \sum_{t=1}^{mN} \lambda(t)\left(\frac{y + Y_t}{L_-} - \frac{y - Y_t}{L_+}\right) \tag{3.44}$$

ここで L_+, L_- はそれぞれ式 (3.45),(3.46) で与えられる。

$$L_+ = (x - X_t)^2 + (y - Y_t)^2 \tag{3.45}$$

$$L_- = (x - X_t)^2 + (y + Y_t)^2 \tag{3.46}$$

3.3.2　2 次元場の表面電荷法プログラムの確認

C++にて 2 次元場の表面電荷法プログラムを作成し、解析解が与えられる電極形状について、表面電荷法で計算した電場を解析解と比較し、プログラムの動作チェックを行った。3 次元の表面電荷法プログラムのチェックと同様に、同軸の二重の円筒電極が形成する電場を計算した。$r_{in} = 10.0$[mm],$r_{out} = 20.0$[mm] の二重円筒に、$V_{in} = -1.0$[V],$V_{out} = 1.0$[V] の電圧を印加したときの動径方向の電場を計算した。表面電荷法で電場を計算した電場と解析解を図 3.16 に示す。また表面電荷法により計算した電場の解析解との誤差を図 3.17 に示す。解析解と表面電荷法で計算した電場が 0.4 %以下で一致しており、2 次元場の表面電荷法プログラムが正常に動作していることを確認した。

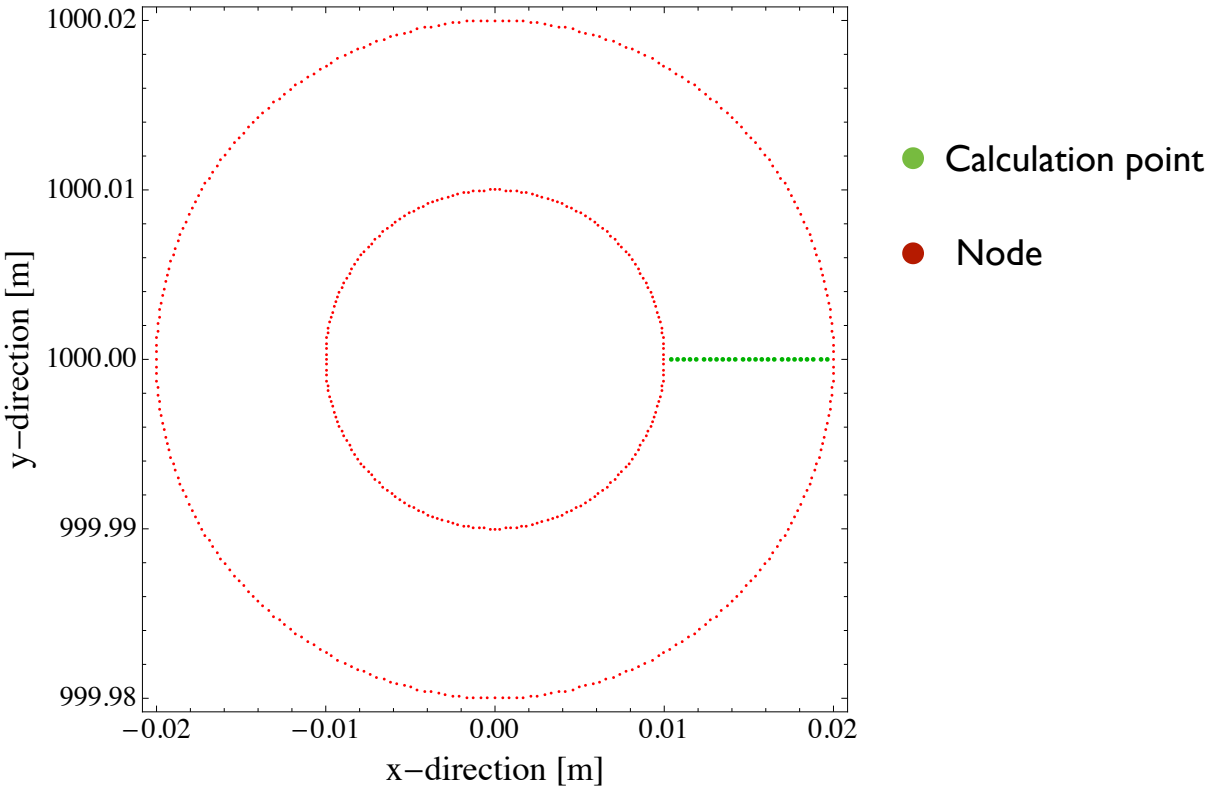


図 3.15: 2 重円筒の模擬（節点のみ表示）と電場の計算点の配置

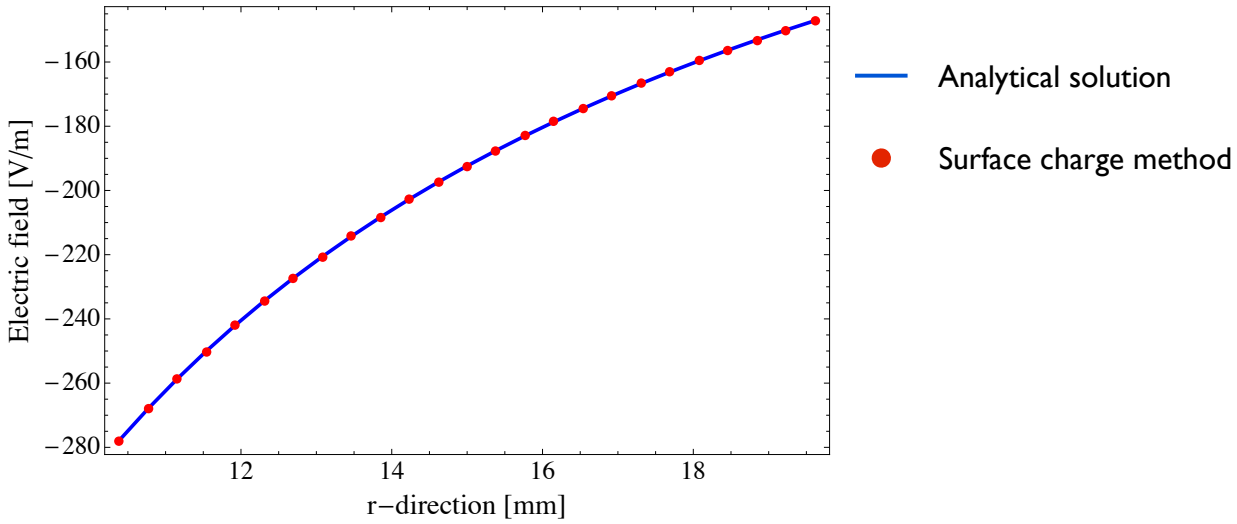


図 3.16: 表面電荷法の計算結果と解析解の比較

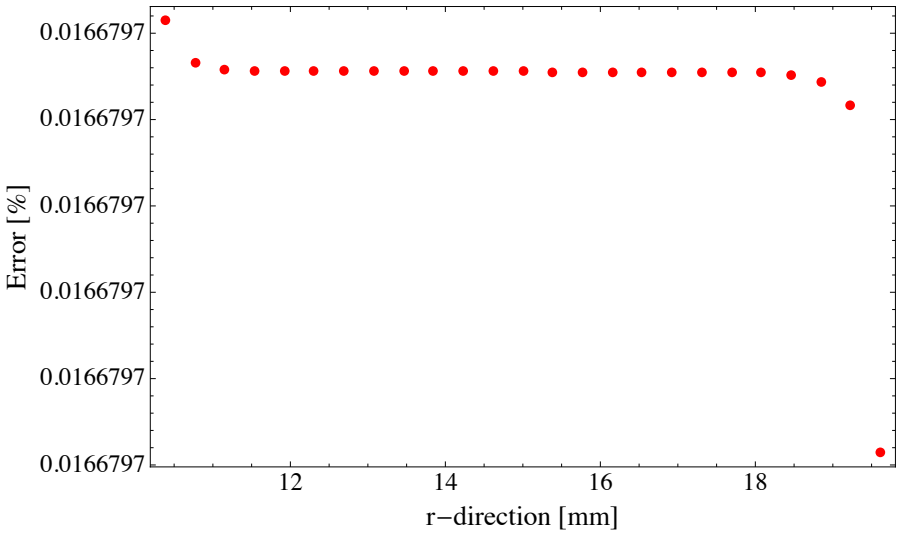


図 3.17: 表面電荷法の誤差

3.4 Fourier 展開による２次元場の計算

２次元場のポテンシャル $\phi(x,y)$ は、Fourier 展開により、極座標で一般に式 (3.47) で表すことができる。

$$\begin{aligned}\phi(x,y) &= \phi(r,\theta) \\ &= \sum_{n=0}^{\infty}\Big(\frac{r}{R}\Big)^n\Big\{C_n cos(n\theta)+D_n sin(n\theta)\Big\} \qquad (3.47)\end{aligned}$$

Fourier 展開係数 C_n, D_n が与えられれば、式 (3.47) を微分することで２次元場の解析的な表現を得ることができる。解析的に電場を表現できれば、表面電荷法での電場の計算のように数値積分の過程がないため、高速に電場を計算でき有用である。さらに、電極形状を変更したときの電場の変化を、Fourier 展開係数を比較すれば定量的に評価できる。

まず、C_n を計算する方法について説明する。式 (3.47) の両辺に $cos(m\theta)$ を乗じて、$-\pi<\theta<\pi$ の範囲で積分する。

$$\begin{aligned}\int_{-\pi}^{\pi}\phi(r,\theta)cos(m\theta)d\theta &= \int_{-\pi}^{\pi}\sum_{n=0}^{\infty}\Big(\frac{r}{R}\Big)^n\Big\{C_n cos(n\theta)+D_n sin(n\theta)\Big\}cos(m\theta)d\theta \\ \int_{-\pi}^{\pi}\phi(r,\theta)cos(n\theta)d\theta &= C_n\Big(\frac{r}{R}\Big)^n\int_{-\pi}^{\pi}cos^2(n\theta)d\theta \qquad (3.48)\end{aligned}$$

ここで式 (3.48) を導くにあたり、公式 (3.49),(3.50) を用いた。

$$\int_{-\pi}^{\pi}sin(n\theta)cos(m\theta)d\theta = 0 \quad (for\ any \quad n,m) \qquad (3.49)$$

$$\int_{-\pi}^{\pi}cos(n\theta)cos(m\theta) = 0 \quad (n\neq m) \qquad (3.50)$$

さらに式 (3.48) には公式 (3.51) を適用することができる。

$$\int_{-\pi}^{\pi}cos^2(n\theta)d\theta=\begin{cases}2\pi & n=0\\ \pi & n\neq 0\end{cases} \qquad (3.51)$$

したがって、式 (3.48) の右辺の $\left(\frac{r}{R}\right)^n\int_{-\pi}^{\pi}cos^2(n\theta)d\theta$ の部分は、R,r,θ を適当に指定すれば任意の n について計算できる。左辺は、数値積分により計算するが、$\phi(r,\theta)$ については２次元の表面電荷法を用いて計算すればよい。左辺は周期関数の数値積分なので、台形公式を用いて計算するのが精度がよい [17]。式 (3.48) の C_n 以外の部分が全て数値的に計算できるので、任意の n について C_n を計算することができる。

D_n の計算する方法について説明する。式 (3.47) の両辺に $sin(n\theta)$ を乗じて、$-\pi < \theta < \pi$ の範囲で積分すると式 (3.52) を得る。

$$\begin{aligned}\int_{-\pi}^{\pi}\phi(r,\theta)sin(m\theta)d\theta &= \int_{-\pi}^{\pi}\sum_{n=0}^{\infty}\Big(\frac{r}{R}\Big)^n\Big\{C_n cos(n\theta)+D_n sin(n\theta)\Big\}cos(n\theta)d\theta \\ \int_{-\pi}^{\pi}\phi(r,\theta)sin(n\theta)d\theta &= D_n\Big(\frac{r}{R}\Big)^n\int_{-\pi}^{\pi}sin^2(n\theta)d\theta \end{aligned} \tag{3.52}$$

式 (3.52) には公式 (3.53) を適用することができる。

$$\int_{-\pi}^{\pi}sin^2(n\theta)d\theta = \begin{cases}0 & n=0\\ \pi & n\neq 0\end{cases} \tag{3.53}$$

したがって、D_n も C_n 同様に計算することができる。

電場の Fourier 展開係数の計算例を示す。断面形状が双曲線の四重極ロッドが形成する電場の Fourier 展開係数をもとめる。図 3.18 に示すように 2 次元の表面電荷法のメッシュを生成し、各々の電極は ±1[V] を印加する。ポテンシャルを計算する点の数は、左辺を数値積分する際の積分点の数に対応する。式 (3.47) で、R をロッドの内接円半径、$r = 0.8R$ とした。C_n, D_n の計算結果を表 3.1 に示すが、Fourier 展開係数の C_2 のみが現れている。この電場の x, y 座標での解析的な形式を求めてみる。式 (3.47) からポテンシャルは式 (3.54) で与えられる。

$$\begin{aligned}\phi(r,\theta) &= C_2\Big(\frac{r}{R}\Big)^2 cos(2\theta) \\ &= C_2\Big(\frac{r}{R}\Big)^2(cos^2\theta - sin^2\theta)\end{aligned} \tag{3.54}$$

さらに式 (3.55),(3.56) で極座標から直交座標に変換する。

$$cos\theta = \frac{x}{r} \tag{3.55}$$

$$sin\theta = \frac{y}{r} \tag{3.56}$$

そして式 (3.57) を得る。

$$\begin{aligned}\phi(r,\theta) &= \phi(x,y) \\ &= C_2\frac{x^2-y^2}{R^2}\end{aligned} \tag{3.57}$$

式 (3.57) は、四重極場のポテンシャルと一致しており、この式を x, y で微分すれば電場が得られる。このようにして、2 次元の電場を解析的な式に変換することができる。

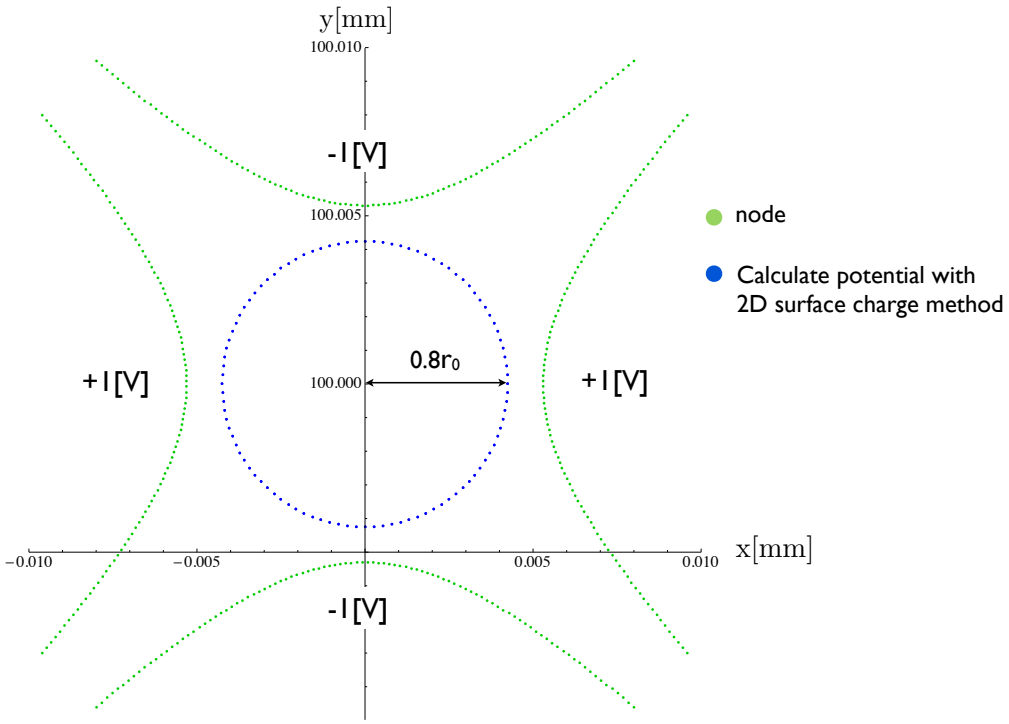


図 3.18: Fourier 展開係数の計算例

C_0=0.000	C_1=0.000	C_2=1.000	C_3=0.000
C_4=0.000	C_5=0.000	C_6=0.000	C_7=0.000
C_8=0.000	C_9=0.000	C_{10}=0.000	C_{11}=0.000
C_{12}=0.000	C_{13}=0.000	C_{14}=0.000	C_{15}=0.000
C_{16}=0.000	C_{17}=0.000	C_{18}=0.000	C_{19}=0.000
D_0=0.000	D_1=0.000	D_2=0.000	D_3=0.000
D_4=0.000	D_5=0.000	D_6=0.000	D_7=0.000
D_8=0.000	D_9=0.000	D_{10}=0.000	D_{11}=0.000
D_{12}=0.000	D_{13}=0.000	D_{14}=0.000	D_{15}=0.000
D_{16}=0.000	D_{17}=0.000	D_{18}=0.000	D_{19}=0.000

表 3.1: Fourier 展開係数の計算結果

3.5　回転対称場の計算

3.5.1　回転対称場の表面電荷法

回転対称場とは、ある座標軸に対して回転対称な形状の電極が形成する電場を意味する。電極形状が回転対称の場合、３次元場の表面電荷法を使用するより、回転対称場の表面電荷法を用いた方がはるかに高速に電場を計算できるため有用である。ここでは回転対称場の表面電荷法について説明する。

図 3.19 に示すように、電極表面の形状は z 軸を回転軸とした回転対称な帯状の要素 (Element) によって模擬する。

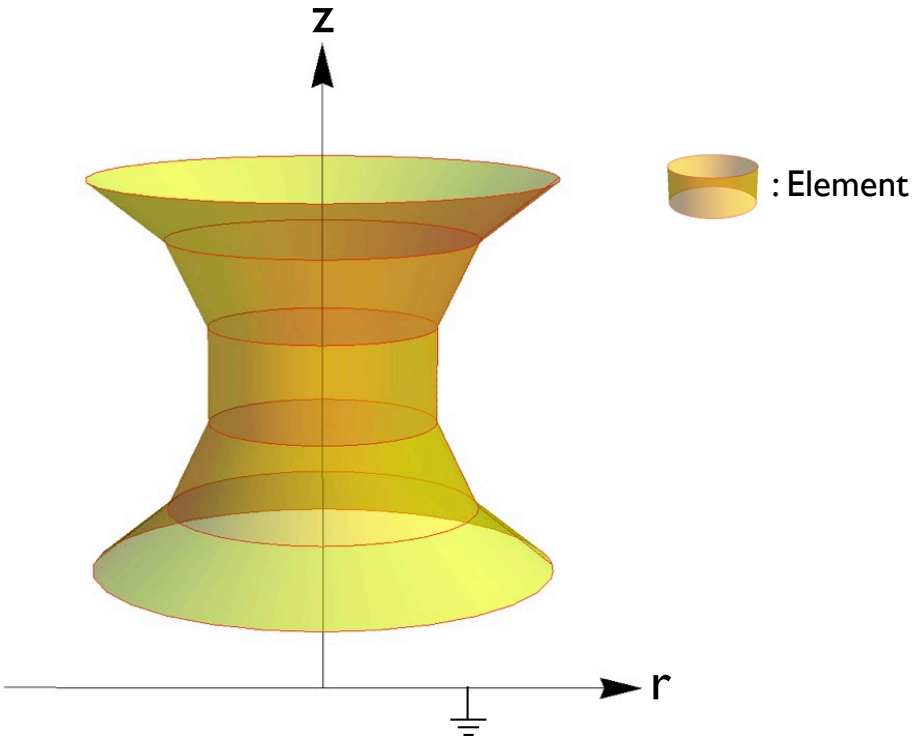


図 3.19: 回転対な帯状の要素による電極の模擬の例

図 3.20 に示すように、要素の断面は線分となり、要素の断面の端を節点 (Node) と呼ぶことにする。各々の要素に番号付けを行い、e 番目の要素を E_e $(e = 1, \cdots, m)$ と表す。節点も番号付けを行い、i 番目の節点を P_i $(i = 1, \cdots, n)$ と表し、その位置ベクトルを (R_i, Z_i)、表面電荷密度を σ_i と表す。

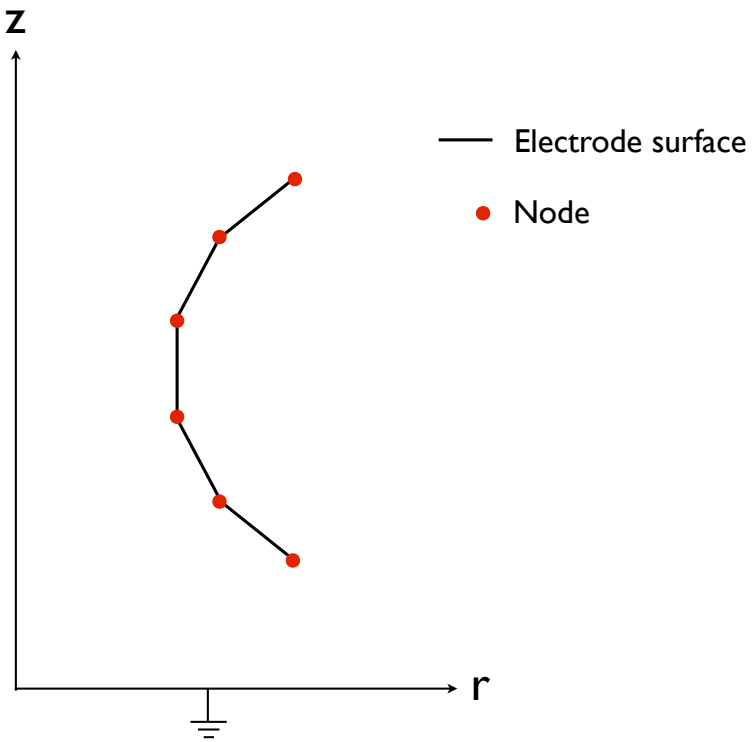


図 3.20: 電極模擬の断面図

要素内部の表面電荷密度は、座標 (図 3.20 で言えば r, z) の 1 次式として与える。このように定めることで、隣あう要素同士で表面電荷密度がなめらかに接続され、ひとつの要素内部で表面電荷密度が一定であるとする場合よりも、電位・電場の計算精度が向上する。まず、要素上の表面電荷密度を座標の 1 次式として与える数学的な表現を導く。

節点 P_i, P_j を含む要素 E_e の表面電荷密度 $\sigma_e(r, z)$ を式 (3.58) のように表す。

$$\sigma_e(r, z) = f_{ie}(r, z)\sigma_i + f_{je}(r, z)\sigma_j \tag{3.58}$$

ここで電極の回転対称性から、表面電荷密度 σ と線電荷密度 λ の間に $\lambda = 2\pi R\sigma$ という関係があるので、式 (3.59) の線電荷密度分布の表現も成立する。以下、線電荷密度の表現の方が簡便なので、要素上の線電荷密度を座標の 1 次式として与える数学的な表現を導くことにする。

$$\lambda_e(r, z) = f_{ie}(r, z)\lambda i + f_{je}(r, z)\lambda j \tag{3.59}$$

r, z は要素上の任意の座標である。要素内部の線電荷密度を座標 (r, z) の 1

次式で与えたいので、f_{ie},f_{je} も (r,z) の 1 次式でとなるように定めなければならない (図 3.21)。

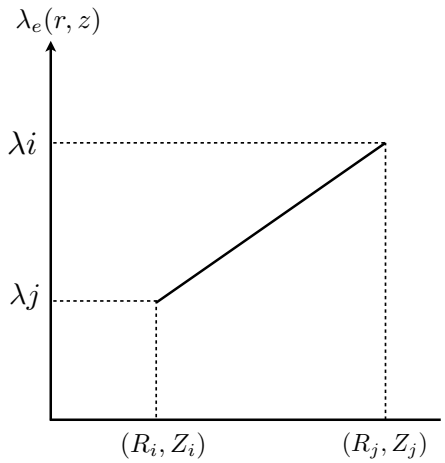


図 3.21: 節点 P_i, P_j を含む要素 E_e 上での線電荷密度の分布

要素 E_e 上の節点 P_i, P_j の線電荷密度 λ_i, λ_j は、f_{ie},f_{je} を用いて式 (3.60) のように表される。

$$\begin{aligned} \lambda_i &= \lambda_e(R_i, Z_i) = f_{ie}(R_i, Z_i)\lambda_i + f_{je}(R_i, Z_i)\lambda_j \\ \lambda_j &= \lambda_e(R_j, Z_j) = f_{ie}(R_j, Z_j)\lambda_i + f_{je}(R_j, Z_j)\lambda_j \end{aligned} \tag{3.60}$$

このとき、f_{ie},f_{je} は式 (3.61) を満たしていなければならない。

$$\begin{aligned} f_{ie}(R_i, Z_i) &= 1 \qquad f_{je}(R_i, Z_i) = 0 \\ f_{ie}(R_j, Z_j) &= 0 \qquad f_{je}(R_j, Z_j) = 1 \end{aligned} \tag{3.61}$$

ここで要素 E_e 上の任意の座標 (r,z) は、媒介変数 u $(-1 < u < 1)$ を導入して式 (3.62) のように表現できる。

$$\begin{aligned} r(u) &= \frac{1-u}{2}R_i + \frac{1+u}{2}R_j \\ z(u) &= \frac{1-u}{2}Z_i + \frac{1+u}{2}Z_j \end{aligned} \tag{3.62}$$

(x,y) を式 (3.62) で定めた場合、$f_{ie}(x,y), f_{je}(x,y)$ は式 (3.63),(3.64) に決まる。

$$f_{ie}\big\{r(u), z(u)\big\} = \frac{1-u}{2} \tag{3.63}$$

$$f_{je}\big\{r(u), z(u)\big\} = \frac{1+u}{2} \tag{3.64}$$

式 (3.63),(3.64) は、式 (3.61) の条件を満たしており、図 3.22 に示すように u に比例して変化するので適切である。

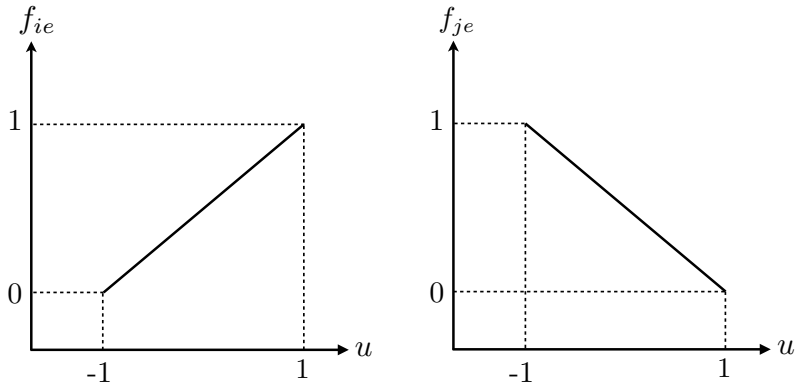


図 3.22: 要素上 $(-1 < u < 1)$ での関数 f_{ie}, f_{je}の値

したがって、電極全体での線電荷密度の分布 $\lambda(r,z)$ は、式 (3.65) で表すことができる。

$$\lambda(r,z) = \sum_{i=1}^{n} \lambda_i \sum_{e=1}^{m} f_{ie}(r,z) \tag{3.65}$$

ただし、要素 E_e が節点 P_i を含まない場合、$f_{ie}(r,z) = 0$ とする。式 (3.65) から電極全体の線電荷密度分布は、節点の線電荷密度が与えられれば、一意に決まることが分かる。そこで、節点の線電荷密度を未知数とした、電極表面の境界条件を表す連立一次方程式を導く。

z 軸を軸として回転対称な、線電荷密度 λ のリング電荷が (R,X) にある場合のポテンシャル分布は式 (3.66) で表される。

$$\begin{aligned} V(r,z) &= 2\pi R\lambda\Big\{\frac{K(k_1)}{\sqrt{(r+R)^2+(z-Z)^2}} - \frac{K(k_2)}{\sqrt{(r+R)^2+(z+Z)^2}}\Big\} \\ &= \lambda F(r,z,R,Z) \end{aligned} \tag{3.66}$$

ここでは真空の誘電率 ε_0 を $\varepsilon_0 = \frac{1}{2\pi^2}$ としている。また K は第 1 種完全楕円積分であり、引数の k_1, k_2 はそれぞれ式 (3.67),(3.68) で与えられる。

$$k_1 = \sqrt{\frac{4rR}{(r+R)^2+(z-Z)^2}} \tag{3.67}$$

$$k_2 = \sqrt{\frac{4rR}{(r+R)^2+(z+Z)^2}} \tag{3.68}$$

第 1 種完全楕円積分の計算法は 3.5.2 に記す。関数 $F(r,z,R,Z)$ は式 (3.69) で定義する。

$$F(r,z,R,Z) = 2\pi R\Big\{\frac{K(k_1)}{\sqrt{(r+R)^2+(z-Z)^2}} - \frac{K(k_2)}{\sqrt{(r+R)^2+(z+Z)^2}}\Big\} \tag{3.69}$$

このポテンシャルの表現は $(R,-z)$ にある鏡像電荷の影響も含むので $z=0$ がアース電位となる。したがって、節点 P_i のポテンシャルが V_i と与えれば式 (3.70) が成り立つ。

$$\begin{aligned} V_i &= V(R_i,Z_i) \\ &= \sum_{j=1}^{n}\lambda_j\sum_{e=1}^{m}\int_{s_e} f_{je}(R,Z)F(R_i,Z_i,R,Z)ds \\ &= \sum_{j=1}^{n}\lambda_j\sum_{e=1}^{m}A_{ije} \\ &= \sum_{j=1}^{n}A_{ij}\lambda_j \end{aligned} \tag{3.70}$$

A_{ije}, A_{ij} はそれぞれ式 (3.71),(3.72) で表される。

$$A_{ije} = \int_{s_e} f_{je}(R,Z)F(R_i,Z_i,R,Z)ds \tag{3.71}$$

$$A_{ij} = \sum_{e=1}^{m}A_{ije} \tag{3.72}$$

A_{ije} は節点 P_j を含む要素のひとつである E_e の表面電荷が、節点 P_i に形成する電位係数である。A_{ij} は節点 P_j を含む全ての要素の表面電荷が節点 P_i に形成する電位係数なので、A_{ije} を足し合わせたものである。

A_{ije} は式 (3.71) を媒介変数 u の積分に変換し、その後ガウスの数値積分公式を適用して計算する。

$$\begin{aligned} A_{ije} &= \int_{s_e} f_{je}(R,Z)F(R_i,Z_i,R,Z)ds \\ &= l_e\int_{-1}^{1} f_{je}(R(u),Z(u))F(R_i,Z_i,R(u),Z(u))du \\ &= l_e\sum_{k=1}^{N} w_k f_{je}\{R(u_k),Z(u_k)\}F\{R_i,Z_i,R(u_k),Z(u_k)\} \end{aligned} \tag{3.73}$$

N はガウスの数値積分の積分点数、u_k は積分点、w_k は重みを表す。l_e は式

(3.74) で与えられる。

$$l_e = \sqrt{(R_i - R_j)^2 + (Z_i - Z_j)^2} \tag{3.74}$$

ここで式 (3.63),(3.64) から $(R(u_k), Z(u_k))$ は、式 (3.75),(3.76) で表される。

$$R(u_k) = \frac{1-u_k}{2} R_i + \frac{1+u_k}{2} R_j \tag{3.75}$$

$$Z(u_k) = \frac{1-u_k}{2} Z_i + \frac{1+u_k}{2} Z_j \tag{3.76}$$

$N = 3$ のガウスの数値積分公式を用いたときの $(R(u_k), Z(u_k))$ を図示すると図 3.23 のようになる。

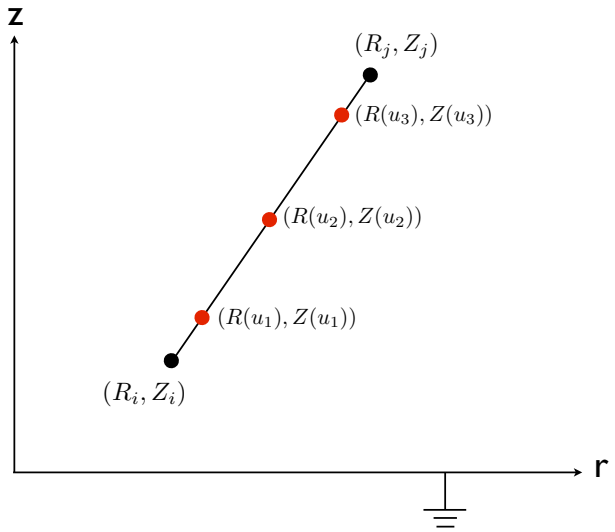


図 3.23: 3 点のガウスの数値積分公式を用いたときの $(R(u_k), Z(u_k))$ の位置

A_{ije} を数値積分で求めることは、図 3.23 に示した $(R(u_k), Z(u_k))$ の位置に、リング電荷（仮想電荷）を配置し、仮想電荷によって E_e の表面電荷の作用を近似していることと同じである。それぞれの仮想電荷の線電荷密度 λ_k は式 (3.77) によって与えられる。

$$\lambda_k = l_e \lambda_i f_{ie}\{R(u_k), Z(u_k)\} + l_e \lambda_j f_{je}\{R(u_k), Z(u_k)\} \tag{3.77}$$

式 (3.72),(3.73) を用い、節点の線電荷密度を未知数とした電極表面の境界条件を表す連立方程式 (3.78) をつくることができる。

$$\begin{bmatrix} A_{11} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \tag{3.78}$$

連立方程式 (3.78) を解き、線電荷密度 λ_1〜λ_n をもとめることができる。

線電荷密度 λ_1〜λ_n が計算できたので、式 (3.65) から全ての要素表面に配置した仮想電荷の電荷量を計算することができる。表面電荷の作用を仮想電荷によって近似しているので、任意の位置に電極が形成する電場は、全ての仮想電荷の寄与をたし上げることで計算でき、式 (3.79),(3.80) のようになる。

$$E_r(r,z)=\sum_{t=1}^{mN}\frac{-\pi R_t\lambda_t}{r}\left[\frac{\left\{R_t^2-r^2+(z-Z_t)^2\right\}E(k_1)-\left\{(r-R_t)^2+(z-Z_t)^2\right\}K(k_1)}{\sqrt{(r+R_t)^2+(z-Z_t)^2}(r-R_t)^2+(z-Z_t)^2}-\right.$$
$$\left.\frac{\left\{R_t^2-r^2+(z+Z_t)^2\right\}E(k_2)-\left\{(r-R_t)^2+(z+Z_t)^2\right\}K(k_2)}{\sqrt{(r+R_t)^2+(z+Z_t)^2}(r-R_t)^2+(z+Z_t)^2}\right] \tag{3.79}$$

$$E_z(r,z)=\sum_{t=1}^{mN}2\pi R_t\lambda_t\left[\frac{(z-Z_t)K(k_1)}{\sqrt{(r+R_t)^2+(z-Z_t)^2}(r-R_t)^2+(z-Z_t)^2}-\right.$$
$$\left.\frac{(z+Z_t)K(k_2)}{\sqrt{(r+R_t)^2+(z+Z_t)^2}(r-R_t)^2+(z+Z_t)^2}\right] \tag{3.80}$$

ここで K は第一種完全楕円積分、E は第二種完全楕円積分であり、これらの計算方法は 3.5.2 に記す。なお、式 (3.79),(3.80) における楕円積分の引数 k_1, k_2 はそれぞれ式 (3.81),(3.82) で与えられる。

$$k_1 = \sqrt{\frac{4rR_t}{(r+R_t)^2+(z-Z_t)^2}} \tag{3.81}$$

$$k_2 = \sqrt{\frac{4rR_t}{(r+R_t)^2+(z+Z_t)^2}} \tag{3.82}$$

3.5.2 第一, 二種完全楕円積分の数値計算

第一種完全楕円積分 $K(k)$ は式 (3.83) で定義される。

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}} \tag{3.83}$$

第二種完全楕円積分 $E(k)$ は式 (3.84) で定義される。

$$E(k) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta \tag{3.84}$$

第一種、二種完全楕円積分は算術幾何平均を利用して計算できる。

正の 2 数 a, b の算術幾何平均は以下の数列を用いて定める。

$$\begin{aligned} a_0 &= \frac{a+b}{2} \\ b_0 &= \sqrt{ab} \end{aligned}$$

を初期値として、

$$\begin{aligned} a_n &= \frac{a_{n-1} + b_{n-1}}{2} \\ b_n &= \sqrt{a_{n-1} b_{n-1}} \\ c_n &= \sqrt{a_n^2 - b_n^2} \end{aligned}$$

を計算していくと、数列 $\{a_n\}$,$\{b_n\}$ は同じ値 $M(a, b)$ に収束し、この $M(a, b)$ が正の 2 数 a, b の算術幾何平均である。

第一種完全楕円積分 $K(k)$ は

$$K(k) = \frac{\pi}{2M(1, \sqrt{1 - k^2})} \tag{3.85}$$

と計算する。

第二種完全楕円積分 $E(k)$ は

$$E(k) = \frac{\pi}{2M(1, \sqrt{1 - k^2})} \left(1 - \sum_{n=0}^{\infty} 2^{n-1} c_n^2\right) \tag{3.86}$$

と計算する。

3.5.3　回転対称場の表面電荷法プログラムの確認

C++にて回転対称場の表面電荷法プログラムを作成し、解析解が与えられる電極形状について、表面電荷法で計算した電場を解析解と比較し、プログラムの動作チェックを行った。3次元の表面電荷法プログラムのチェックと同様に、同軸の二重の円筒電極が形成する電場を計算した。$r_{in} = 1.0$[mm],$r_{out} = 2.0$[mm] の二重円筒に、$V_{in} = -1.0$[V],$V_{out} = 1.0$[V] の電圧を印加したときの動径方向の電場を計算した。表面電荷法で電場を計算した電場と解析解を図 3.25 に示す。また表面電荷法により計算した電場の解析解との誤差を図 3.26 に示す。解析解と表面電荷法で計算した電場が 0.4 %以下で一致しており、回転対称場の表面電荷法プログラムが正常に動作していることを確認した。

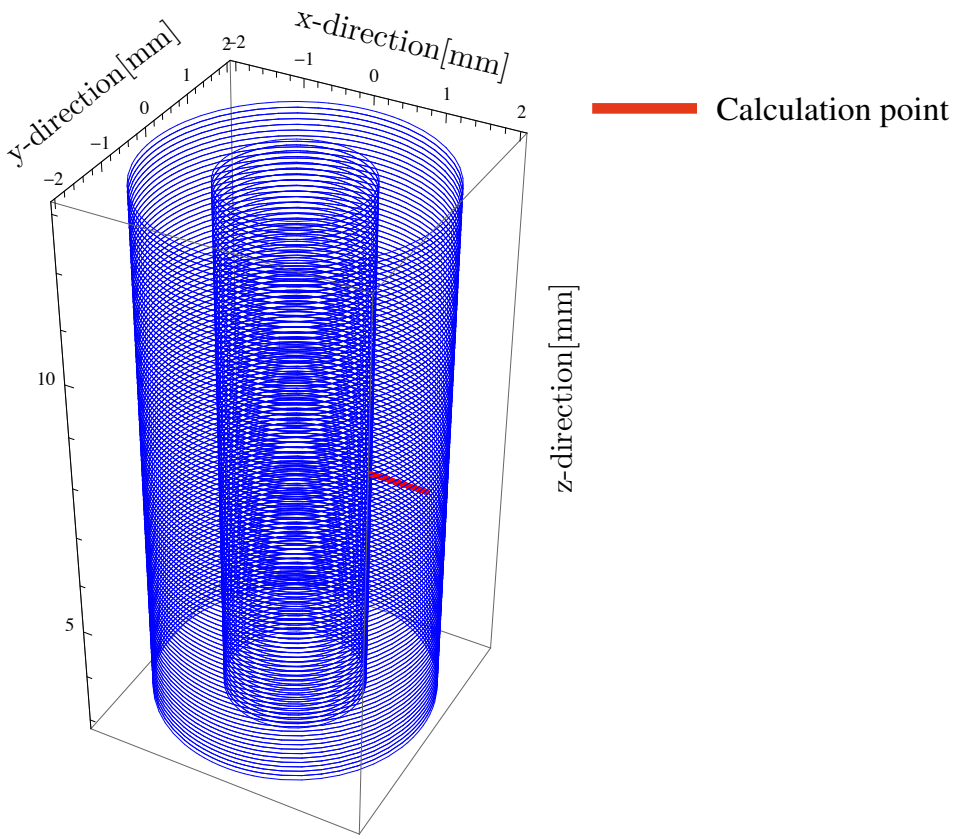


図 3.24: 2 重円筒の模擬（節点のみ表示）と電場の計算点の配置

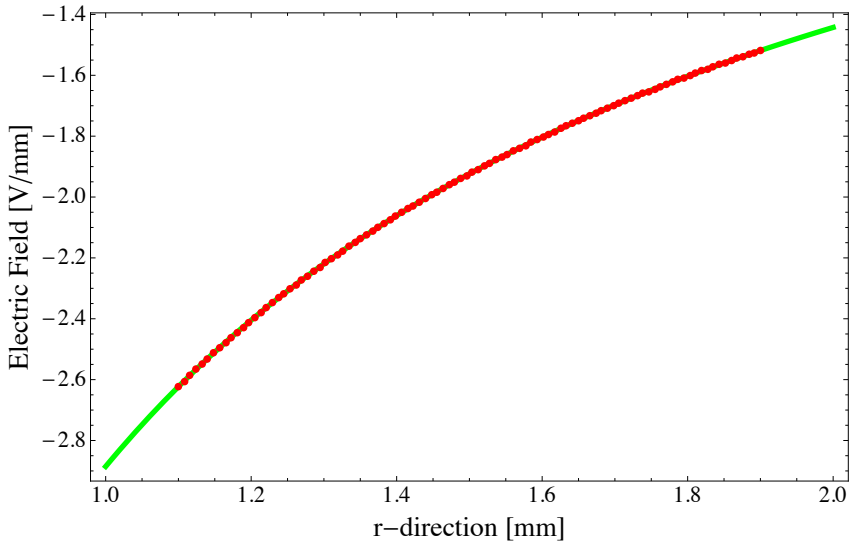


図 3.25: 表面電荷法の計算結果と解析解の比較

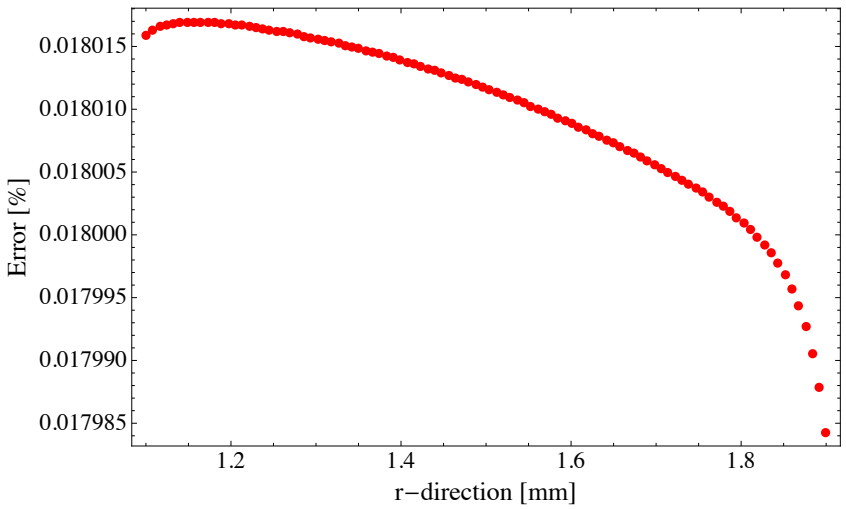


図 3.26: 表面電荷法の誤差

3.6 イオン軌道の計算

電場 $\boldsymbol{E}$ の中での質量 m、電荷 q のイオンの運動方程式は式 (3.87) で与えられる。

$$m\frac{d^2\boldsymbol{r}}{dt^2} = q\boldsymbol{E} \tag{3.87}$$

ここで $\boldsymbol{r}$ はイオンの軌道、t は時間である。イオン軌道 $\boldsymbol{r}$ は式 (3.87) の運動方程式を解くことで求めることができる。しかし、電場 $\boldsymbol{E}$ が複雑になれば、解析的にイオン軌道 $\boldsymbol{r}$ をもとめることは困難である。そこで、式 (3.87) を数値的に積分して、イオン軌道 $\boldsymbol{r}$ を求めることを考える。

イオンの運動方程式に４次のルンゲ・クッタ法を適用するには、式 (3.87) を式 (3.88),(3.89) の２つの一階常微分方程式に書き換えればよい。

$$\frac{d\boldsymbol{v}}{dt} = \frac{q\boldsymbol{E}}{m} \tag{3.88}$$

$$\frac{d\boldsymbol{r}}{dt} = \boldsymbol{v} \tag{3.89}$$

ここで $\boldsymbol{v}$ はイオンの速度を表す。一階常微分方程式なので、４次のルンゲ・クッタ法を適用することができる。x, y, z のそれぞれの方向に分けて計算する。x 方向について、関数 f, g を定義する。

$$f(t, x, v_x) = v_x \tag{3.90}$$

$$g(t, x, v_x) = \frac{q}{m}E_x \tag{3.91}$$

計算アルゴリズムは、h を微小時間として式 (3.92)〜(3.101) のようになる。

$$kf1 = h \times f(t, x, v_x) \tag{3.92}$$

$$kg1 = h \times g(t, x, v_x) \tag{3.93}$$

$$kf2 = h \times f\Big(t + \frac{h}{2}, x + \frac{kf1}{2}, v_x + \frac{kg1}{2}\Big) \tag{3.94}$$

$$kg2 = h \times g\Big(t + \frac{h}{2}, x + \frac{kf1}{2}, v_x + \frac{kg1}{2}\Big) \tag{3.95}$$

$$kf3 = h \times f\Big(t + \frac{h}{2}, x + \frac{kf2}{2}, v_x + \frac{kg2}{2}\Big) \tag{3.96}$$

$$kg3 = h \times g\Big(t + \frac{h}{2}, x + \frac{kf2}{2}, v_x + \frac{kg2}{2}\Big) \tag{3.97}$$

$$kf4 = h \times f\Big(t + \frac{h}{2}, x + \frac{kf3}{2}, v_x + \frac{kg3}{2}\Big) \tag{3.98}$$

$$kg4 = h \times g\Big(t + \frac{h}{2}, x + \frac{kf3}{2}, v_x + \frac{kg3}{2}\Big) \tag{3.99}$$

微小時間 h 後の位置 x、速度 v_x はそれぞれ

$$x(t+h) = x(t) + \frac{kf1 + 2kf2 + 2kf3 + kf4}{6} \tag{3.100}$$

$$v(t+h) = v(t) + \frac{kg1 + 2kg2 + 2kg3 + kg4}{6} \tag{3.101}$$

y 方向、z 方向も同様に計算できる。

3.7　イオンとガス分子の衝突

大気圧イオン源により生成されたイオンは、オリフィス 1,2 とイオンガイド 1,2 を経てリニアイオンイオントラップまで輸送される (図 2.5)。そして、リニアイオントラップへ入射したイオンをトラップするには、イオンをトラップ内の緩衝ガスと衝突させて運動エネルギーを落とす必要がある。

したがって、リニアイオントラップ内でのイオンの運動状態をシミュレーションするには、衝突を考慮する必要がある。衝突は確率的な事象であるため、疑似乱数を用いたモンテカルロシミュレーションが適している。以下に衝突の計算手法を説明する。

衝突確率

イオンとガスとの衝突確率は平均自由行程の概念を利用して計算した。平均自由行程は、ガスの圧力・温度・形状、イオンの速度・形状から決定される。多数のガス粒子の運動は、ガスが熱平衡状態にある場合、Maxwell-Boltzmann の速度分布に従う。ガス粒子の速さ v を

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{3.102}$$

とすると、質量 m のガス粒子の速さが v と $v + \Delta v$ の間にある確率は

$$f(v)\Delta v = 4\pi v^2 \left(\frac{m}{2\pi k_B T}\right)^{3/2} exp\left(\frac{-mv^2}{2k_B T}\right)\Delta v \tag{3.103}$$

で表される。確率が最大となるイオンの速さ v_m は

$$v_m = \sqrt{\frac{2k_B T}{m}} \tag{3.104}$$

である。平均速度 $\bar{v}$ は

$$\bar{v} = \int_0^\infty v f(v) dv = \frac{2}{\sqrt{\pi}} v_m \tag{3.105}$$

となる。

また、ガスの平均自由行程が λ であるとき、このガス粒子が飛距離 x で衝突する確率 $P(x)$ は

$$P(x) = 1 - exp\left(-\frac{x}{\lambda}\right) \tag{3.106}$$

で表される。$0 \leq A < 1$ の一様乱数 A を発生させ、$A < P(x)$ なら衝突したと判定する。

平均自由行程

平均自由行程 λ は下式を用いた。

$$\lambda = \frac{\bar{V}}{\bar{V_r}\sigma n} \tag{3.107}$$

$$\sigma = \pi(R+r)^2 \tag{3.108}$$

$$\bar{V_r} = \sqrt{\bar{V}^2 + \bar{v}^2} \tag{3.109}$$

ここで R：イオン半径、r：ガス半径、$\bar{V}$：イオンの平均速度、$\bar{v}$：ガスの平均速度、$\bar{v_r}$：ガスから見たイオンの平均相対速度、n：ガスの単位体積あたり粒子数、σ：衝突断面積である。平均相対速さ $\bar{V_r}$ の表式だが、$\bar{V} \gg \bar{v}$ ではガスが全て停止しているモデル、$\bar{v} \simeq \bar{v'}$ ではイオンとガスが Maxwll-Boltzmann 分布しているとするモデルで導いた $\bar{V_r}$ に一致する [18]。

衝突後のイオンの速度

ガスと衝突後のイオンの速度の計算法について説明する。図 3.27 に示すように、イオンの中心点と衝突点を結ぶ方向の単位ベクトルを $\boldsymbol{e_i}$、これに直交する単位ベクトルを $\boldsymbol{e_j}, \boldsymbol{e_k}$ とする ($\boldsymbol{e_i} = \boldsymbol{e_j} \times \boldsymbol{e_k}$)。衝突前のイオンの速度 $\boldsymbol{V}$、ガス粒子の速度 $\boldsymbol{v}$ は

$$\boldsymbol{V} = V_i\boldsymbol{e_i} + V_j\boldsymbol{e_j} + V_k\boldsymbol{e_k} \tag{3.110}$$

$$\boldsymbol{v} = v_i\boldsymbol{e_i} + v_j\boldsymbol{e_j} + v_k\boldsymbol{e_k} \tag{3.111}$$

のように各単位ベクトルの成分に分解できる。衝突後の速度はそれぞれ $\boldsymbol{V'}$、$\boldsymbol{v'}$ とする。なめらかな衝突を仮定すれば、$\boldsymbol{e_j}, \boldsymbol{e_k}$ 方向の速度は変化しないので、式 (3.112) が成立する。

$$v'_j = v_j \qquad V'_j = V_j \qquad v'_k = v_k \qquad V'_k = V_k \tag{3.112}$$

さらに衝突が剛体球の完全弾性衝突であるとすれば式 (3.113) が成立する。

$$\frac{v'_i - V'_i}{v_i - V_i} = -1 \tag{3.113}$$

また $\boldsymbol{e_i}$ 方向の運動量保存則より式 (3.114) が成立する。

$$MV_i + mv_i = MV'_i + mv'_i \tag{3.114}$$

式 (3.113), (3.114) を整理して、衝突後の $\boldsymbol{e_i}$ 方向のイオンとガス粒子の速度を表す、式 (3.115), (3.116) を得る。

$$V_i' = \frac{2m}{M+m} v_i + \frac{M-m}{M+m} V_i \tag{3.115}$$

$$v_i' = \frac{m-M}{M+m} v_i + \frac{2M}{M+m} V_i \tag{3.116}$$

また、なめらかな衝突を仮定したので、式 (3.112) が示すように衝突の前後で $\boldsymbol{e_j}, \boldsymbol{e_k}$ 方向の速度は変化しない。したがって、衝突後の速度 $\boldsymbol{V'}$ は式 (3.117) で表される。

$$\boldsymbol{V'} = (V_i' - V_i)\boldsymbol{e_i} + \boldsymbol{V} \tag{3.117}$$

したがって、$\boldsymbol{e_i}$ が決まれば衝突後のイオンの速度を計算できる。

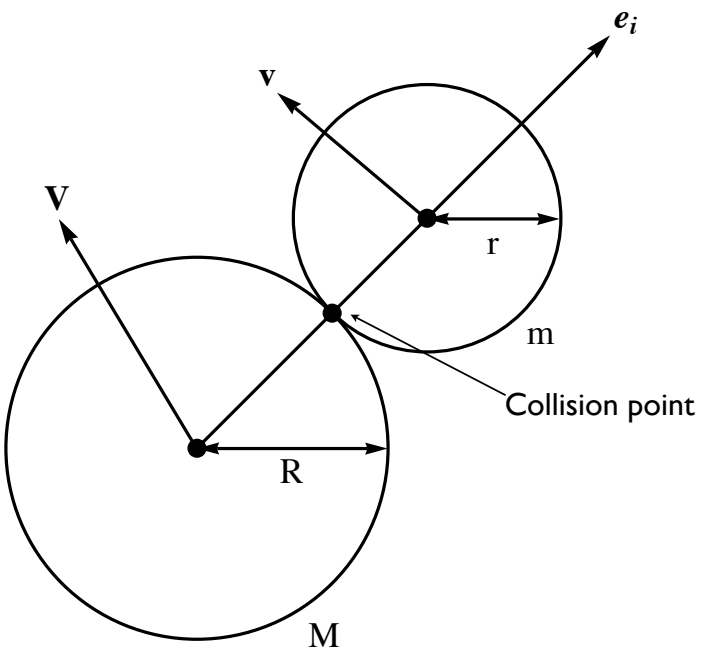


図 3.27: イオンとガス粒子の衝突

つづいて、イオンの中心と衝突点を結ぶ方向の単位ベクトル $\boldsymbol{e_i}$ を決定する方法について説明する。イオンとガス粒子の相対速度方向の単位ベクトル $\boldsymbol{e_\alpha}$ は $\boldsymbol{V}, \boldsymbol{v}$ の定義から、式 (3.118) で表される。

$$\boldsymbol{e_\alpha} = \frac{\boldsymbol{V} - \boldsymbol{v}}{|\boldsymbol{V} - \boldsymbol{v}|} \tag{3.118}$$

また $\boldsymbol{e_\alpha}$ と直交する単位ベクトル $\boldsymbol{e_\beta}$, $\boldsymbol{e_\gamma}$ を $\boldsymbol{e_\alpha} = \boldsymbol{e_\beta} \times \boldsymbol{e_\gamma}$ を満たすように適当に定める。また、ガス粒子の速度 $\boldsymbol{v}$ は Maxwell-Boltzmann 分布に従うように、Box-Muller 法によりランダムに決定する。

状況を整理すると、ガス粒子が静止した系では、図 3.28 のようになる。イオンに衝突するガス粒子は、イオンの進行方向の前面 (図 3.28 で灰色に着色されたイオンの面) のどこかに衝突する。

次にイオンとガス粒子の衝突点を決める。半径 1 の半球上の点 (X_i, Y_i, Z_i) を式 (3.119),(3.120) に従ってランダムに生成する。

$$X_i^2 + Y_i^2 < 1 \tag{3.119}$$

$$Z_i = \sqrt{1 - (X_i^2 + Y_i^2)} \tag{3.120}$$

$\boldsymbol{e_\alpha}, \boldsymbol{e_\beta}, \boldsymbol{e_\gamma}$ を基底に用いた座標での衝突点は、イオンの半径 R をもちいて (rX_i, rY_i, rZ_i) と定める。これは、図 3.28 のイオンの進行方向の前面のある１点を、ランダムに決定したことになっている。このように衝突点を定めることで、正面衝突から、かするような衝突までを表現できる。

衝突点が指定されたので、イオンの中心と衝突点を結ぶ方向の単位ベクトル $\boldsymbol{e_i}$ は式 (3.121) と決定される。

$$\boldsymbol{e_i} = Z_i \boldsymbol{e_\alpha} + X_i \boldsymbol{e_\beta} + Y_i \boldsymbol{e_\gamma} \tag{3.121}$$

$\boldsymbol{e_i}$ が決定されれば、式 (3.117) により、ガスと衝突後のイオンの速度を決定できる。

$\boldsymbol{e_\beta}, \boldsymbol{e_\gamma}$ は $\boldsymbol{e_\beta}$, $\boldsymbol{e_\gamma}$ を $\boldsymbol{e_\alpha} = \boldsymbol{e_\beta} \times \boldsymbol{e_\gamma}$ を満たすように任意に定めてよいと述べたが、例えば式 (3.122)〜(3.124) のようにして定めてもよい。

$$\boldsymbol{e_\beta} = \frac{\boldsymbol{e_x} \times \boldsymbol{e_\alpha}}{|\boldsymbol{e_x} \times \boldsymbol{e_\alpha}|} \tag{3.122}$$

$$\boldsymbol{e_x} = (1, 0, 0) \tag{3.123}$$

$$\boldsymbol{e_\gamma} = \boldsymbol{e_\alpha} \times \boldsymbol{e_\beta} \tag{3.124}$$

また、モンテカルロ法での乱数生成は全て Mersenne Twister 法 [19] を用いた。

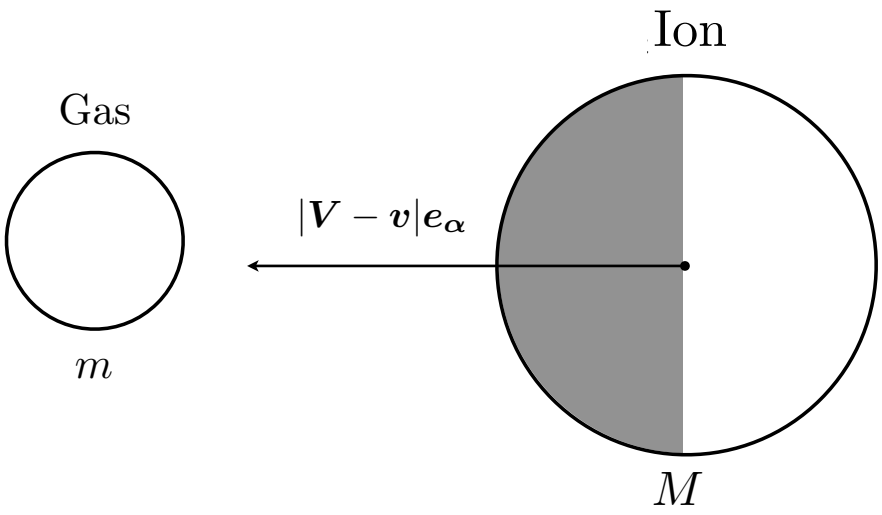


図 3.28: ガス粒子が静止した系での衝突の模式図

イオンとガス分子の衝突計算プログラムの確認

温度 T で熱平衡にあるガス分子中を、イオンが飛行する状況を想定する。イオンとガス粒子を剛体球とみなし、完全弾性衝突するとすれば、イオンが単位時間あたりに受ける力 F はイオンの速度 V の関数として解析的に与えられて式 (3.125)〜(3.127) で表される [20]。

$$F(V) = -p\pi(R+r)^2\left\{\frac{e^{-x_0^2}}{\sqrt{\pi}x_0}(1+2x_0^2)+(2x_0^2+2-\frac{1}{2x_0^2})\Phi(x_0)\right\} \tag{3.125}$$

$$\Phi(x_0) = \frac{2}{\sqrt{\pi}}\int_0^{x_0} e^{-x^2}dx \tag{3.126}$$

$$x_0 = V\sqrt{\frac{m}{2k_BT}} \tag{3.127}$$

イオンの運動方程式

$$M\frac{dV(t)}{dt} = F(V) \tag{3.128}$$

を離散化して

$$V_{n+1} = V_n + \frac{1}{M}F(V_n)\Delta t \tag{3.129}$$

を得る。式 (3.129) を用いれば、。式 (3.129) により計算した衝突の冷却効果と、モンテカルロシミュレーションによって計算した冷却効果（イオン 100 個の計算結果を平均）を比較した。各種パラメータは、$m = 40$[u], $M = 195$[u],$r = 3.41$[Å],$R = 10$[Å], イオンの初期エネルギー $= 30$[eV],$T = 300$[K], ガスの圧力=1.0×10^{-3}[Pa]、である。

冷却過程のわずかなズレは、イオンの平均自由行程を式 (3.109) で近似的に表現しているためと考えられる。しかしながら、衝突による冷却効果の本質は損なわれおらず、妥当な計算手法なであると判断した。

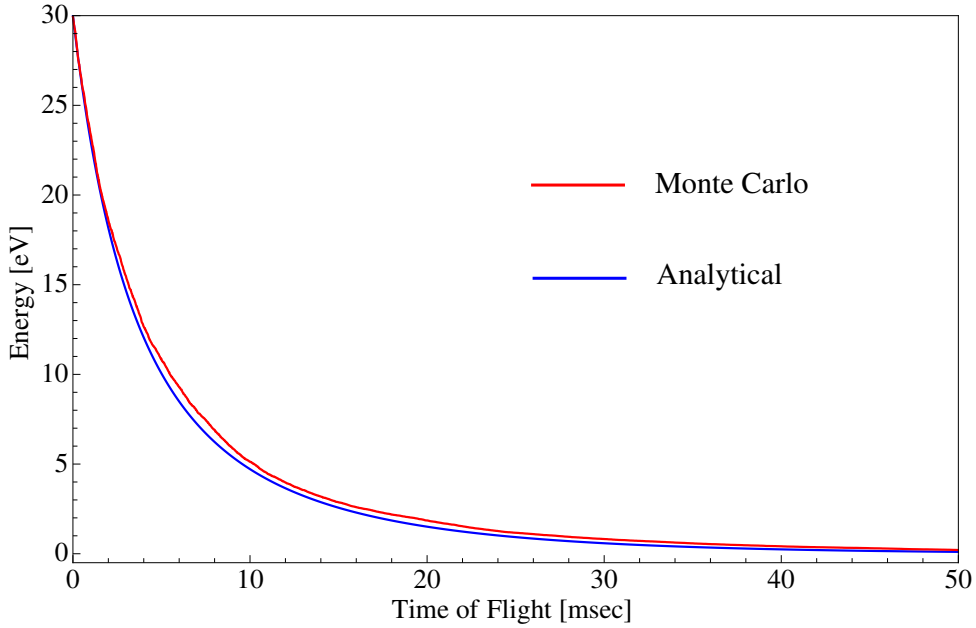


図 3.29: 衝突による冷却効果

第4章　イオンガイド中のイオン軌道シミュレーション

ロッド電極間に板状電極を挿入したリニアイオントラップに捕獲されるイオンの空間分布・運動状態を知ること、さらに蓄積されたイオン排出する際の時間収束性・空間収束性を評価することを目的にイオン軌道シミュレーションを行った。リニアイオントラップに捕獲されるイオンの空間分布・運動状態を正確に知るには、イオンガイドからリニアイオントラップに打ち込まれるイオンの運動状態を知る必要がある。そのために、まずイオンガイド１からイオンガイド２でのイオン軌道のシミュレーションを行った。次にイオンガイド２からリニアイオントラップに打ち込まれたイオンのイオン軌道シミュレーションを行い、蓄積されたイオンの空間分布・運動状態を調べた。また、イオンの蓄積効率を改善するための、リニアイオントラップの電極形状を最適化を行った。そして、リニアイオントラップに蓄積されたイオンを排出するシミュレーションを行った。

以下のシミュレーションでは図 4.1 に示したように、イオンの進行方向を z 方向、z 方向に直交する方向を x, y 方向と定める。

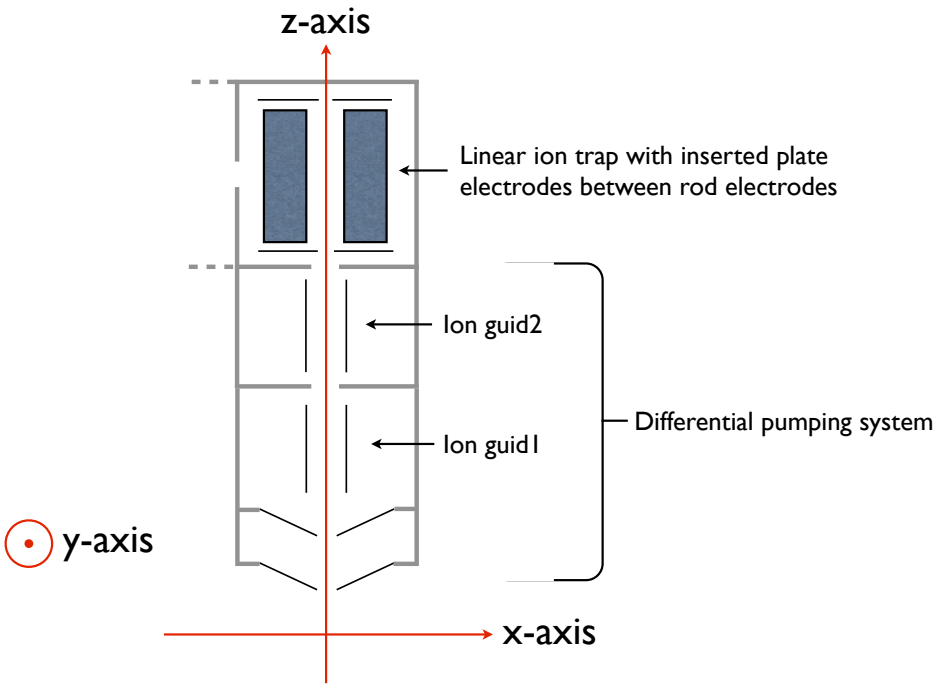


図 4.1: イオンガイド中のイオン軌道シミュレーションでの x, y, z 方向の定義

4.1 四重極場を安定に通過できるイオンの運動条件

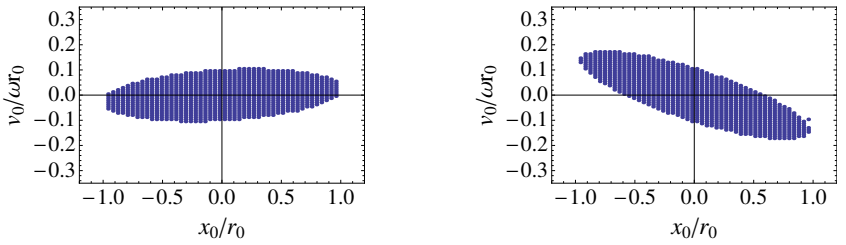
イオンガイドは四重極電場を形成することで、イオンを x, y 方向にトラップし、z 方向に安定に輸送する。大気圧イオン源で生成したイオンは様々な運動状態をもってイオンガイドに入射するが、イオンガイドを安定に通過できる入射条件は限られてるはずである。そこで、まず四重極電場中をある軌道半径内で通過できるイオンの入射条件を調べた。四重極場は x-y 方向で独立なので、１次元のシミュレーションを行った。電場は以下のように解析的に定めた。

$$E_x = -\{U + Vcos(\omega t + \theta_0)\}\frac{2x}{r_0^2} \tag{4.1}$$

ここで r_0 はイオンガイドの内接円半径で $r_0 = 1.0$[mm], $\omega/2\pi = 1.0$[MHz], θ_0 は RF の初期位相である。マシューパラメータは、イオンの x-y 方向の軌道半径が最も小さくなる、$a = 0.0,\ q = 0.58$ とした [21]。このようにマシューパラメータを定めた理由は、イオンガイド１でのイオンの x-y 方向の軌道半径が小さくなれば、イオンガイド２へ安定に導入されるイオン量が多くなると推測したからである。

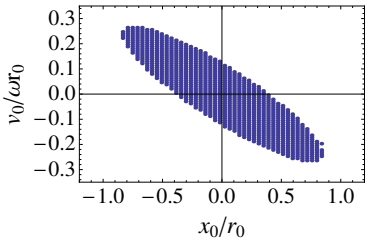
シミュレーション結果を図 4.2 に示す。横軸は式 (4.1) でのイオンの x 方向の初期位置 x_0 を、イオンガイドの内接円半径 r_0 でノーマライズしたものである。また、縦軸はイオンの x 方向の初期速度 v_0 を、四重極場の角周波数 ω とイオンガイドの内接円半径 r_0 との積でノーマライズしたものである。図 4.2 から、四重極場を安定に通過可能なイオンの初期条件は、RF の初期位相 θ_0 に依存していることが分かる。

また、図 4.2 をよく見ると RF の初期位相 θ_0 に依存せずに、イオンガイド１を安定に通り抜けることができるイオンの初期条件が存在する。図 4.3 には、RF の初期位相に依存せずイオンが安定に四重極場を通過できる初期条件と、RF の初期位相が合えばイオンが安定に四重極場を通過できる初期条件を示した。y 方向の運動は、y 方向の電場は x 方向の電場から初期位相が π ずれているだけなので、x 方向の運動の初期位相 θ_0 を π ずらしたものになっているはずである。したがって、y 方向についても、RF の初期位相に依存せずイオンが安定に四重極場を通過できる初期条件は、x 方向と全く同様である。

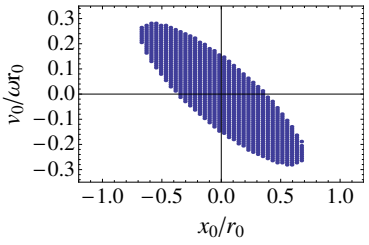


(a) θ_0 =0.0 rad

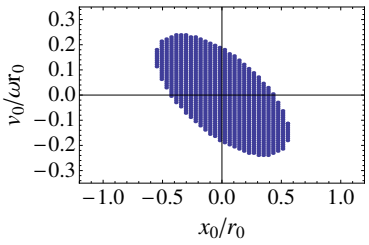
(b) θ_0 =0.2π rad



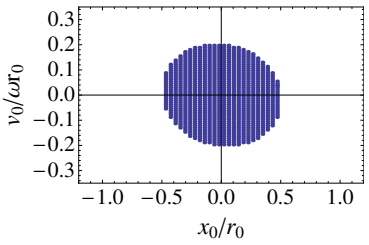
(c) θ_0 =0.4π rad



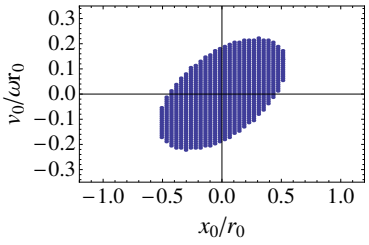
(d) θ_0 =0.6π rad



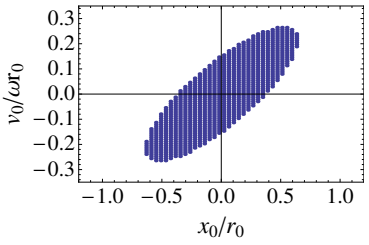
(e) θ_0 =0.8π rad



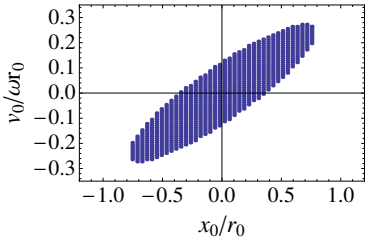
(f) θ_0 =1.0π rad



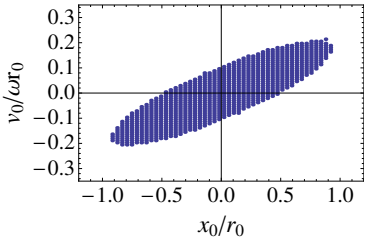
(g) θ_0 =1.2π rad



(h) θ_0 =1.4π rad



(i) θ_0 =1.6π rad



(j) θ_0 =1.8π rad

図 4.2: 四重極場を安定に通過できる初期条件

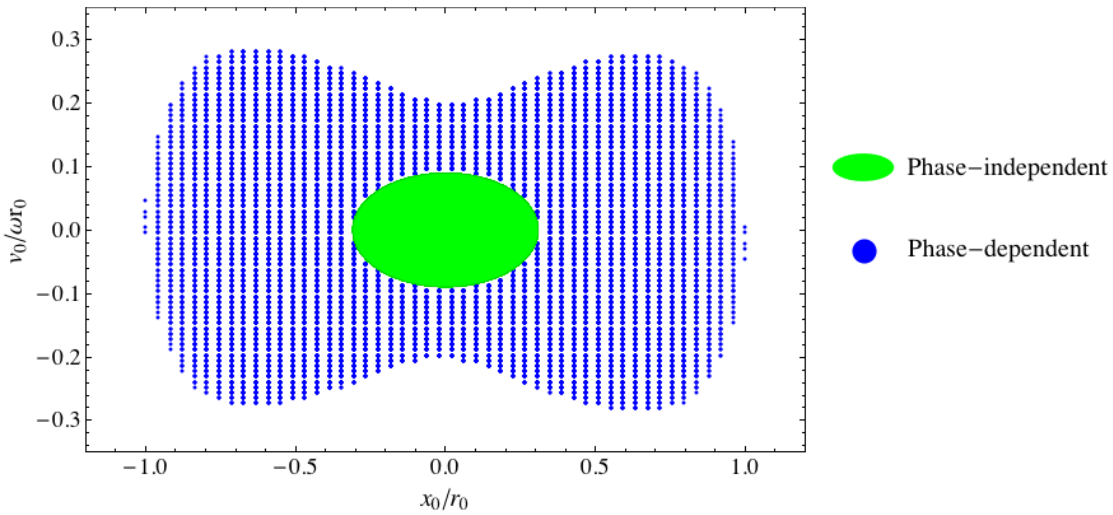


図 4.3: 四重極場を通過できる初期条件

4.2 イオンガイドでのイオン軌道シミュレーション

つづいて、イオンガイド１内をイオンが輸送されていく過程のシミュレーションを行った。イオンガイド１は差動排気系の途中に位置し、真空度は 0.1[Pa] 程度で、イオンの進行方向と直交する方向 (x, y 方向) の運動は残留ガスとの衝突により冷却されていくと考えられる。ただし、z 方向 の運動は、イオンは気流とイオン間のクーロン相互作用により輸送されると考え、イオンの z 方向 の運動エネルギーは一定であると仮定した。

一般的な大気圧イオン源と飛行時間質量分析計を直交加速法により接続した装置では、大気圧イオン源から差動排気系を介して導入されるイオンの進行方向の運動エネルギーは、数十 eV 程度になっている。ここではイオンの z 方向 の運動エネルギーは 30[eV] とし、電場は x, y, z 方向をそれぞれ式 (4.2),(4.3),(4.4) のように定めた。

$$E_x = -(U + V cos\omega t)\frac{2x}{r_0^2} \tag{4.2}$$

$$E_y = (U + V cos\omega t)\frac{2y}{r_0^2} \tag{4.3}$$

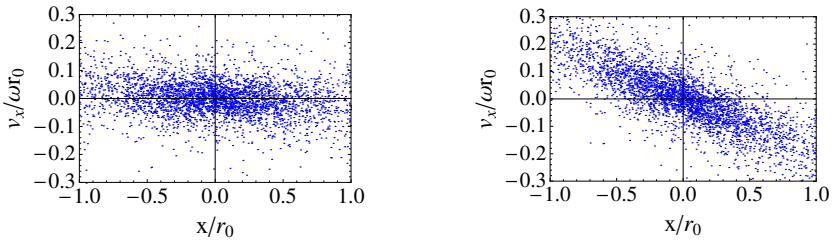
$$E_z = 0 \tag{4.4}$$

ここで r_0 はイオンガイドの内接円半径で $r = 1$[mm], $\omega/2\pi = 1.0$[MHz]、マシューパラメータは $a = 0.0,\ q = 0.58$ とした。このようにマシューパラメータを定めた理由は、イオンガイド１でのイオンの軌道半径が小さくなれば、イオンガイド２へ安定に導入されるイオン量が多くなると推測したからである。イオンの初期条件は、図 4.3 のイオンが RF の初期位相に依存せず四重極場を安定に通過できる初期条件の範囲内に 1 万個ランダムに分布させた。イオンの質量 85[u], 半径 10[Å]、イオンガイド内のガスの質量 40[u], 半径 0.98[Å], 温度 300[K], 圧力 0.1[Pa] とした。イオンガイド中には様々な種類の残留ガスがあると考えられるが、アルゴンで代表させた。イオンの飛行時間はイオンガイドの長さとイオンの z 方向のエネルギーから約 100[μs] とした。また、シミュレーションでのルンゲクッタ法の時間幅 h は RF 周期の 1/30 倍とし、十分になめらかにイオン軌道を再現する。

計算結果を図 4.4, 図 4.5 に示す。図 4.4 の横軸は、イオンがイオンガイド内部を約 100[μs] 後の、x 方向の位置をイオンガイドの内接円半径 r_0 でノーマライズしたものである。縦軸は、イオンがイオンガイド内部を約 100[μs] 後の、イオンの x 方向の速度を RF の角周波数 ω とイオンガイドの内接円半径

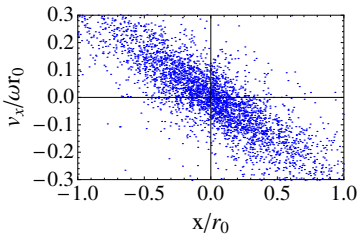
r_0 との積でノーマライズしたものである。x 方向の運動状態・空間分布は RF の位相 ωt に依存しており、イオン群は様々な初期条件で条件でイオンガイドに入射するが、残留ガスとの衝突で x 方向のエネルギーを徐々に失い、運動状態が揃えられていくためであると考えられる。図 4.5 はイオンの y 方向の運動状態・空間分布について表しており、横軸・縦軸の見方は図 4.4 と同様である。y 方向の運動は、概ね x 方向の運動の位相 ωt を π ずらしたものになっており、これは y 方向の電場は x 方向の電場から初期位相が π ずれているためであると考えられる。

また、イオンガイド 2 の真空度は 0.01[Pa] 程度で、イオンと残留ガスの衝突確率は低いので、イオンガイド 2 の中での運動は概ね図 4.4,4.5 の運動を保っていると考えられる。さらに、リニアイオントラップでのイオンの一般的なイオン蓄積時間は ms 程度であり、イオンガイド 2 の RF の周期 μs 程度と比較して圧倒的に長いので、図 4.4, 4.5 に示したような分布のイオンがリニアイオントラップに打ち込まれると考えられる。

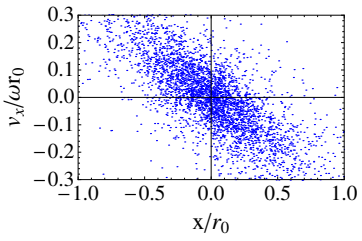


(a) ωt =0.0 rad

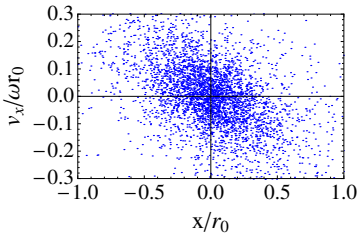
(b) ωt =0.2π rad



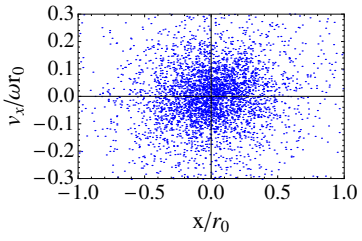
(c) ωt =0.4π rad



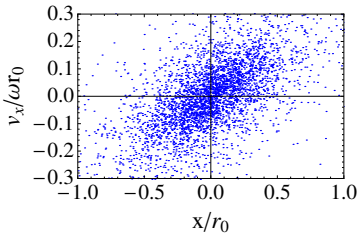
(d) ωt =0.6π rad



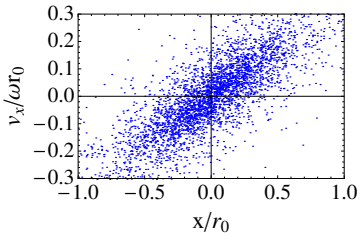
(e) ωt =0.8π rad



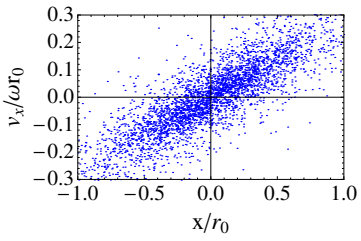
(f) ωt =1.0π rad



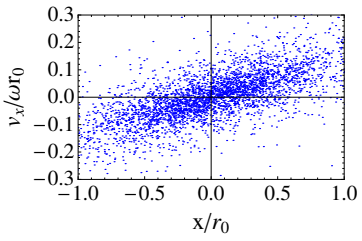
(g) ωt =1.2π rad



(h) ωt =1.4π rad

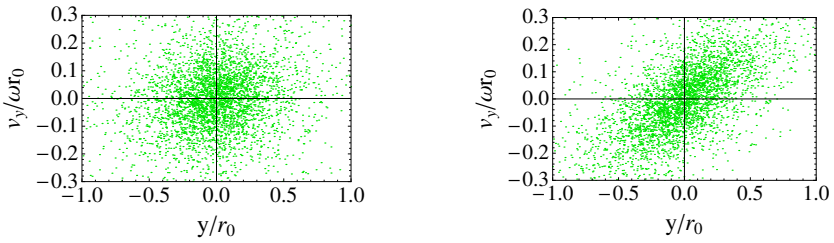


(i) ωt =1.6π rad



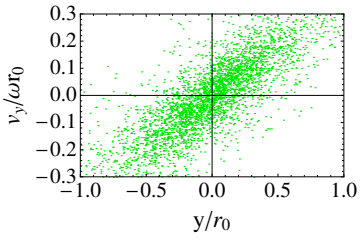
(j) ωt =1.8π rad

図 4.4: イオンガイド 1 を約 100[μs] 飛行したイオンの x 方向の位相空間分布

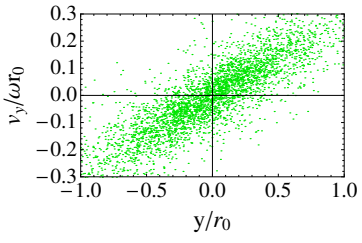


(a) ωt =0.0 rad

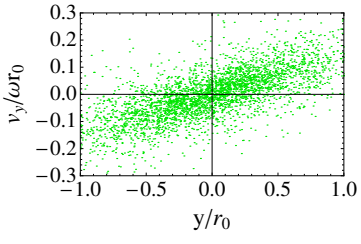
(b) ωt =0.2π rad



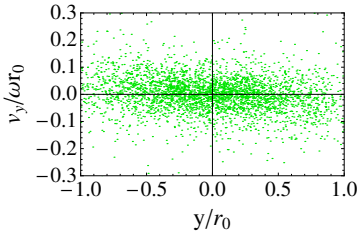
(c) ωt =0.4π rad



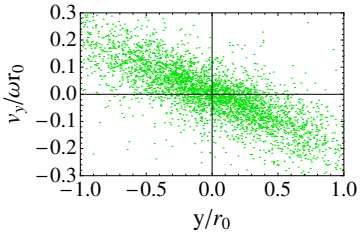
(d) ωt =0.6π rad



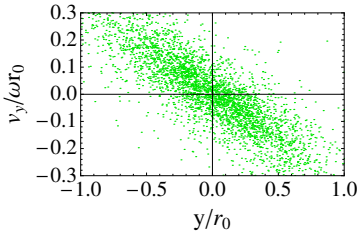
(e) ωt =0.8π rad



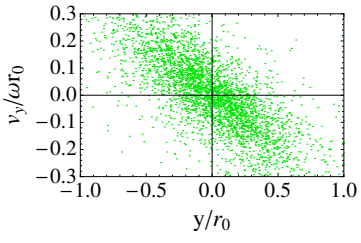
(f) ωt =1.0π rad



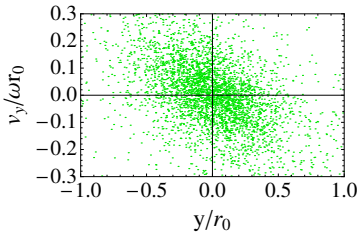
(g) ωt =1.2π rad



(h) ωt =1.4π rad



(i) ωt =1.6π rad



(j) ωt =1.8π rad

図 4.5: イオンガイド 1 を約 100[μs] 飛行したイオンの y 方向の位相空間分布

第5章　リニアイオントラップへのイオンの打ち込み・捕獲シミュレーション

イオンガイドを経て、イオンはリニアイオントラップに打ち込まれる。リニアイオントラップに打ち込まれたイオンは、トラップ内部でバッファーガスと衝突し、運動エネルギーを落とすことによって蓄積される。蓄積されたイオンはさらにバッファーガスとの衝突を繰り返し、冷却されていく。冷却後のイオンがリニアイオントラップ内部でどのような運動状態・空間分布をしているか調べるため、イオンガイドからイオントラップにイオンを打ち込むシミュレーションを行った。

5.1 リニアイオントラップ内部の真空度

シミュレーションを行うにあたり、リニアイオントラップ内部の適切な真空度を、平均自由行程の概念を利用した単純なモデルでおおまかに見積もった。

リニアイオントラップにイオンをトラップするためには、イオンガイドから打ち込まれたイオンがリニアイオントラップを通過するまでに、最低1回はバッファーガスと衝突してエネルギーを落とさなければならない。したがって、イオンをトラップするという観点では真空度は高い方が有利である。

一方で、リニアイオントラップにトラップされたイオンをトラップ軸と直交する方向に排出する際には、イオンはバッファーガスと衝突しないことが望ましい。イオンを排出するという観点では真空度は低い方が有利である。

平均自由行程の概念を利用すれば、イオンがリニアイオントラップに入射するときバッファーガスと衝突する確率 f_1、イオンがリニアイオントラップから排出されるときバッファーガスと衝突しない確率 f_2 は、真空度 P の関数として表現できる。よって、最適な真空度は $f_1(P) \times f_2(P)$ を最大化するような真空度 P である。

f_1 は、 質量 40[u], 半径 0.98[Å], 温度 300[K], 圧力 P のガス中を、質量 85[u], 半径 10[Å], エネルギー 30[eV] のイオンが、リニアイオントラップのトラップ軸方向の長さ $L_{trap} = 28$[mm] を飛行する間にガスと衝突する確率とした (図 5.1)。

$$f_1(P) = 1 - exp\left\{-\frac{L_{trap}}{\lambda_{in}(P)}\right\} \tag{5.1}$$

ここで $\lambda_{in}(P)$ は圧力 P でのイオンの平均自由行程で、式 (3.107)〜(3.109) から導出される。

f_2 は、平均自由行程はイオンの速度に依存するので、リニアイオントラップからイオンを排出するときの速度変化を大まかに考慮した。イオンは一様電場中を、トラップ軸から引き出し電極までの距離 $L_{pull} = 11.5$[mm] 運動するとした (図 5.2)。トラップ軸から引き出し電極までの飛行時間を微小時間 Δt に区切り、時間番号を i とする。Δt の運動のうちに衝突しない確率 $\Delta f(P, i)$ は

$$\Delta f(P, i) = exp\left\{-\frac{\Delta L(i)}{\lambda_{out}(P, V(i))}\right\} \tag{5.2}$$

$$\Delta L(i) = V(i)\Delta t \tag{5.3}$$

$$V(i) = a\,\Delta t \tag{5.4}$$

ここで $\Delta L(i)$：時間番号 i でのイオンの変位、$V(i)$：時間番号 i でのイオンの速度である。a は一様電場中でイオンが受ける加速度である。$\lambda_{out}(P, V(i))$ は $P, V(i)$ に対応する平均自由行程であり、式 (3.107)〜(3.109) から導出される。$f_2(P)$ は $\Delta f(P, i)$ の総乗となり

$$f_2(P) = \prod_{i=0}^{N} \Delta f(P, i) \tag{5.5}$$

である。ここで N は時間の分割数を表し、$N = 100$ とした。

関数 $f_1(P) \times f_2(P)$ の計算結果を図 5.3 に示す。圧力 $P = 10^n$[Pa] として、n と $f_1(P) \times f_2(P)$ の関係をプロットしている。この計算結果から、リニアイオントラップ内の圧力 P は 0.1[Pa] 程度が妥当であると判断した。

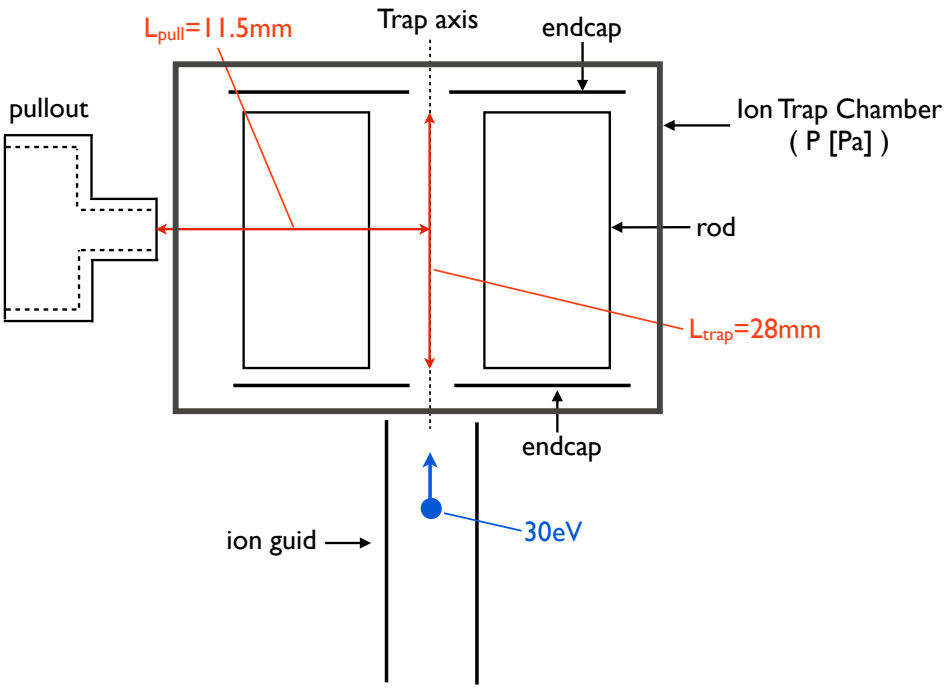


図 5.1: イオントラップ、引き出し電極の位置関係

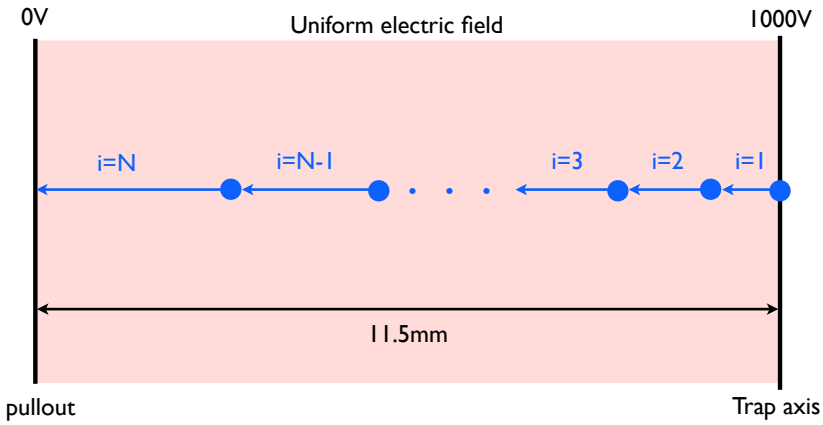


図 5.2: トラップ軸から引き出し電極までのイオンの運動

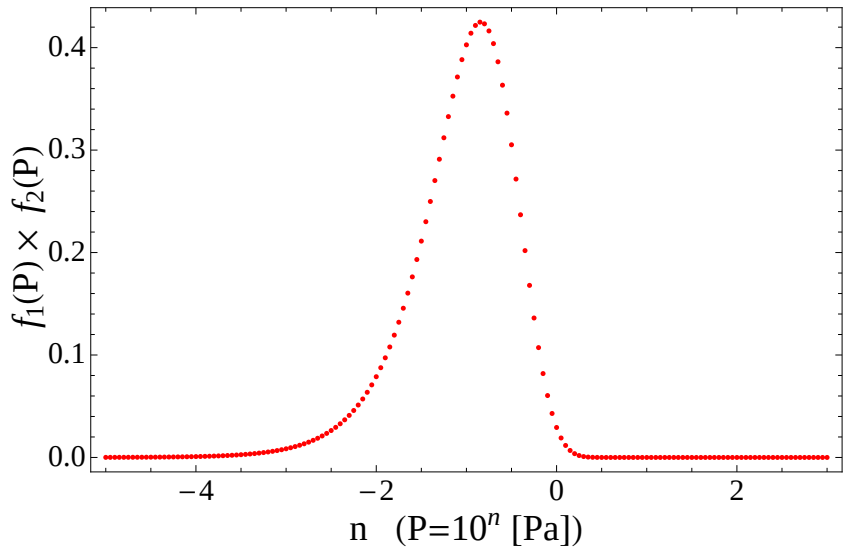


図 5.3: イオントラップ内部の圧力 P と関数 $f_1(P) \times f_2(P)$ の関係

5.2 リニアイオントラップへのイオンの打ち込み・捕獲シミュレーション

イオンガイドからリニアイオントラップに打ち込まれたイオンがトラップされるには、バッファーガスとの衝突が少なくとも 1 回は必要である。したがって、3.4 で説明したような確率的に衝突を起こさせる方法は、バッファーガスとの衝突によるイオンの冷却過程を調べるシミュレーションでは、試行回数が膨大になり能率的ではない。

そこで、イオントラップ内の任意の位置 $z = Z_{rand}$ を乱数で決め、イオントラップに入射したイオンとバッファーガスとの初回の衝突は、$z = Z_{rand}$ で強制的に発生させることにした。2 回目以降の衝突は 3.4 で説明したように確率的に衝突を発生させ、イオンの冷却過程を調べた。図 5.4 にリニアイオントラップへのイオンの打ち込み・捕獲シミュレーションの概念図を示す。イオンガイドの出口から 4[mm] 手前の位置から、イオンを打ち込んだ。このような位置からイオンを打ち込むことで、イオンガイドの端電場の影響を考慮した。イオンの x, y 方向の初期条件は、図 4.4,4.5 に示したような位相空間分布から、ランダムに 10000 個データを抽出し、それを用いた。z 方向には、4.2 で述べたように、30[eV] の初期速度を持たせた。イオンの x 方向の初期条件を図 5.5 に示す、この図では r_0 はイオンガイドの内接円半径である。

シミュレーションの諸条件は以下のとおり。イオンガイドの RF 周波数：$\omega_g/2\pi = 1.0$[MHz](ω_g はイオンガイドの角周波数)、Mathieu パラメータ：$a = 0,\ q = 0.57$、リニアイオントラップの RF 周波数：$\omega/2\pi = 1.0$[MHz](ω はリニアイオントラップの角周波数)、Mathieu パラメータ：$a = 0,\ q = 0.407$、エンドキャップ電圧：29[V]、イオンの質量：85[u]、半径：10[Å]、バッファーガスの質量：40[u]、半径：0.98[Å]、温度：300[K]、圧力：0.1[Pa] とした。イオンガイドとリニアイオントラップの RF の初期位相は同じとし、約 0.5[ms] 飛行させた。イオンの初期条件は、トラップ軸と直交する方向には図 5.5 の位相空間分布をし、z 方向に 30[eV] の運動エネルギーをもっているとした。イオン打ち込み・捕獲シミュレーションでの電場を計算するための表面電荷法のメッシュを図 5.6 に示す。挿入電極は無視し、リニアイオントラップの電極形状は双曲線状であるとしている。

イオン 10000 個を打ち込んだシミュレーション結果を図 5.7〜5.9 に示す。これらの図においては、r_0 とはリニアイオントラップの内接円半径を指す。

イオンは 7110 個トラップされた。x-y 方向ではイオンは RF の位相に依った位相空間分布をしているが、z 方向の位相空間分布はほとんど変化していない。x, y 方向の運動については、イオンガイド中でのシミュレーションのとき同様、様々な初期条件をもったイオンをリニアイオントラップが入射しても、バッファーガスと衝突することによりエネルギーを徐々に失って運動状態が揃えられてゆき、ロッドの RF 電圧に支配された運動をするようになると考えられる。z 方向の運動は、やはりバッファーガスとの衝突で徐々にエネルギーを失ってゆき、z 方向の井戸型ポテンシャルの中で振動運動していると考えられる。

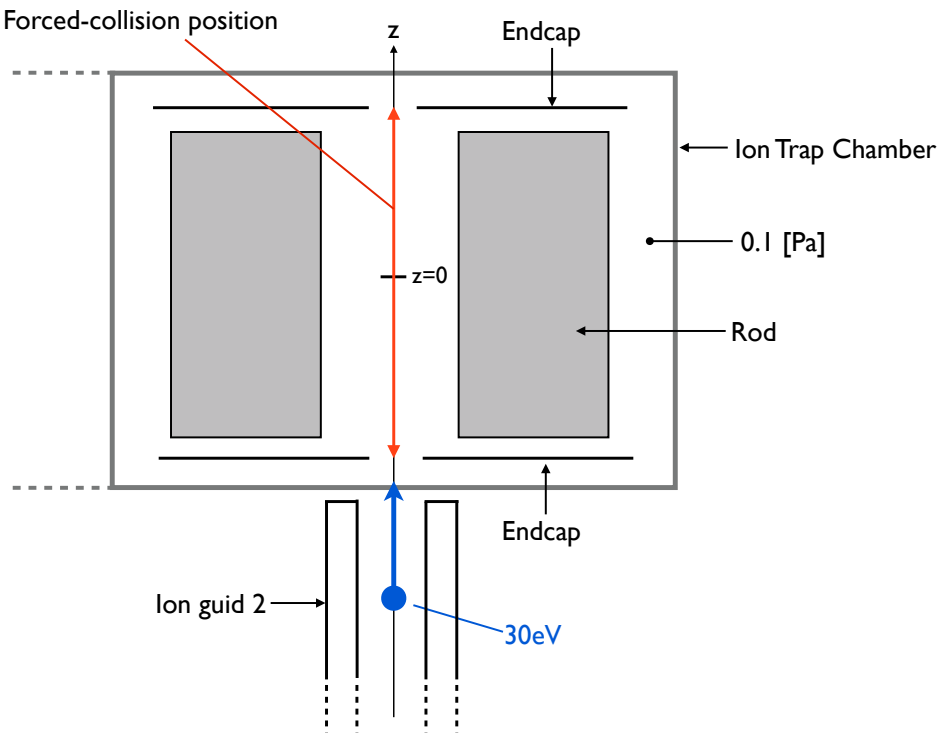


図 5.4: イオン打ち込みシミュレーション概念図

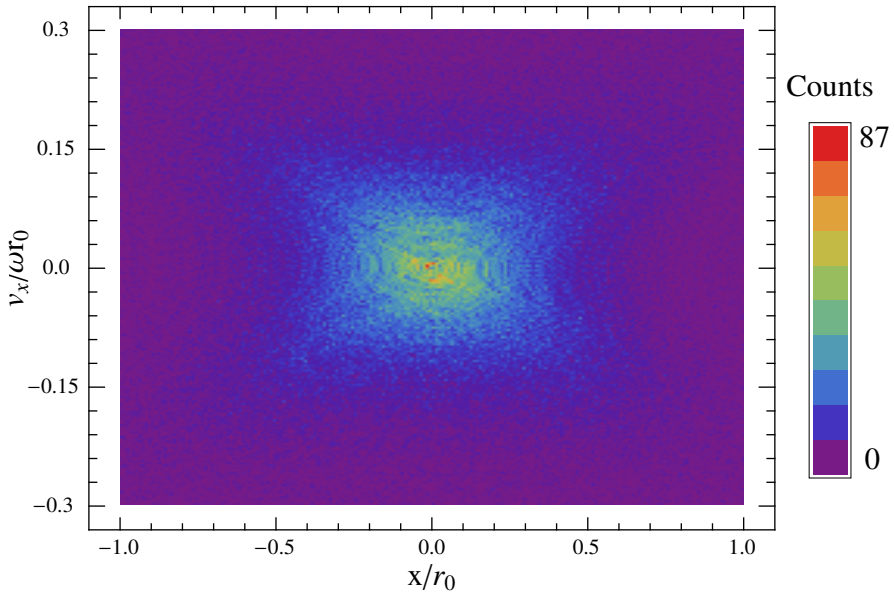


図 5.5: イオン打ち込み・捕獲シミュレーションの x 方向の初期条件

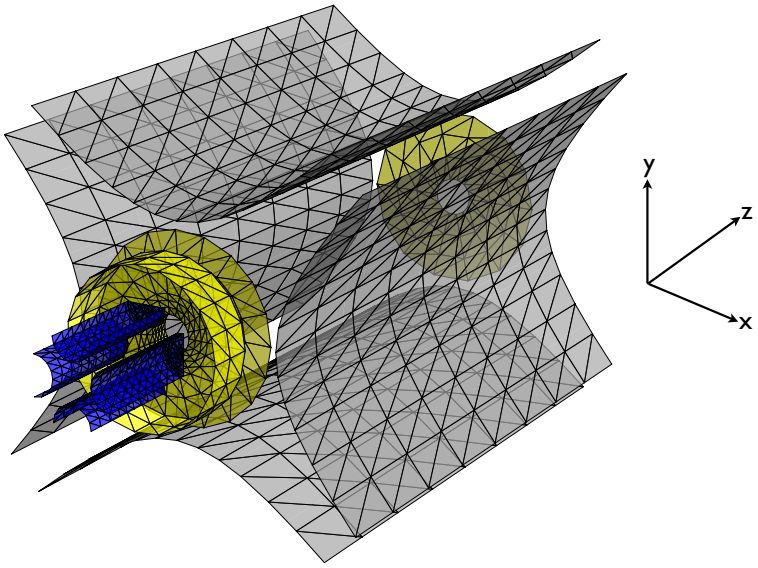
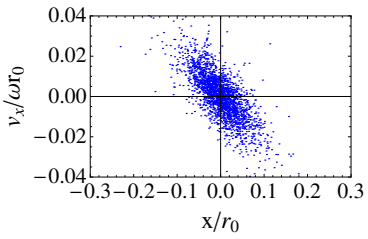
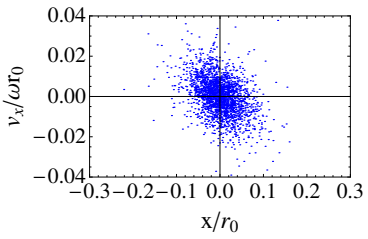


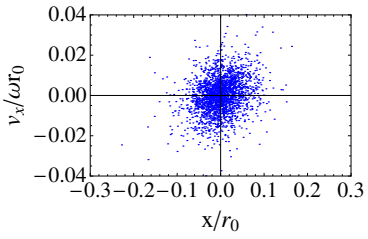
図 5.6: イオン打ち込みシミュレーションのための表面電荷法のメッシュ



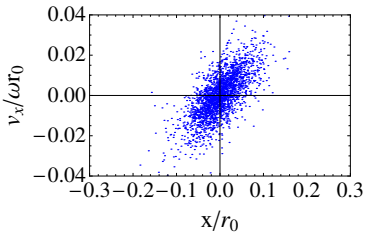
(a) $\omega t = -17/15\pi$ rad



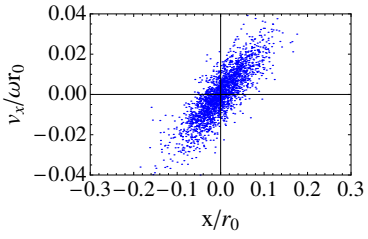
(b) $\omega t = -14/15\pi$ rad



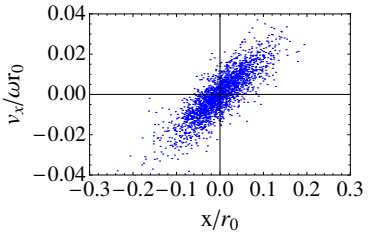
(c) $\omega t = -11/15\pi$ rad



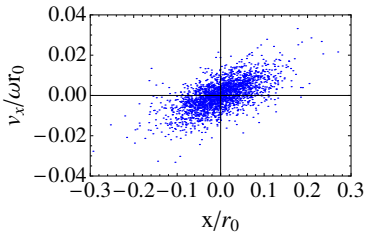
(d) $\omega t = -8/15\pi$ rad



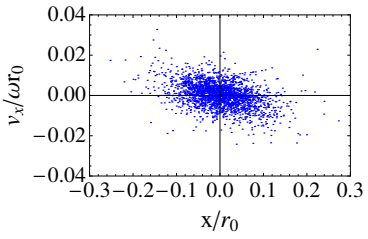
(e) $\omega t = -5/15\pi$ rad



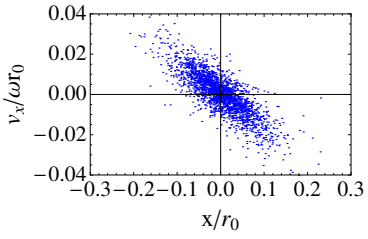
(f) $\omega t = -2/15\pi$ rad



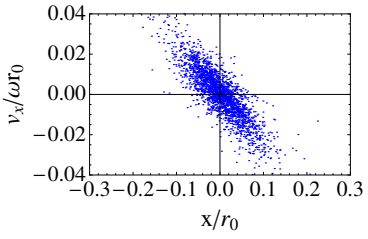
(g) $\omega t = 1/15\pi$ rad



(h) $\omega t = 4/15\pi$ rad

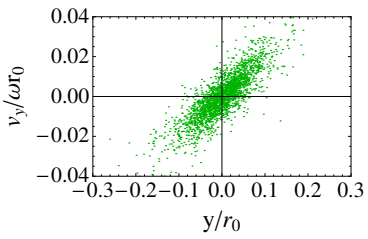


(i) $\omega t = 7/15\pi$ rad

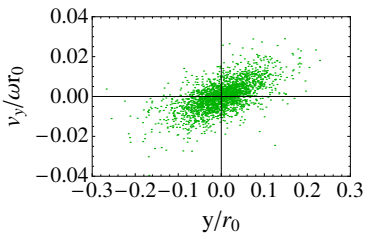


(j) $\omega t = 10/15\pi$ rad

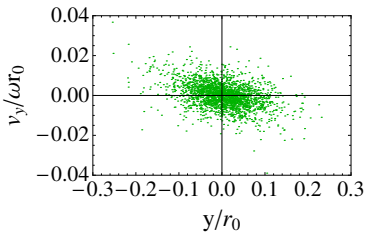
図 5.7: 冷却されたあとの位相空間分布 (x 方向)



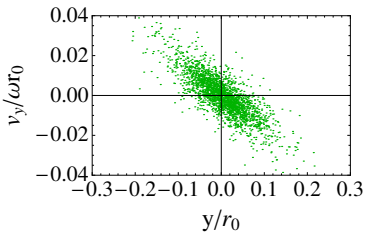
(a) $\omega t = -17/15\pi$ rad



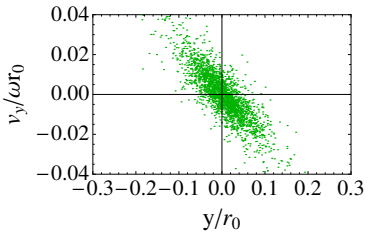
(b) $\omega t = -14/15\pi$ rad



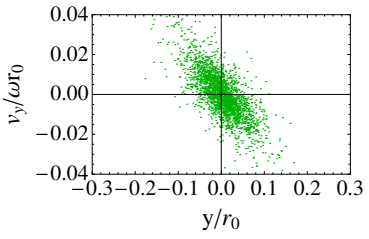
(c) $\omega t = -11/15\pi$ rad



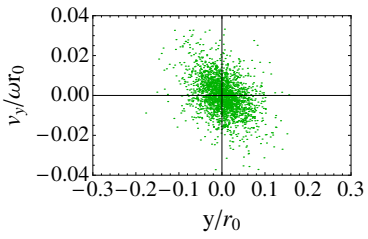
(d) $\omega t = -8/15\pi$ rad



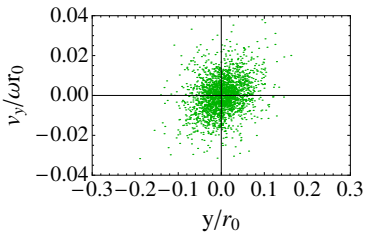
(e) $\omega t = -5/15\pi$ rad



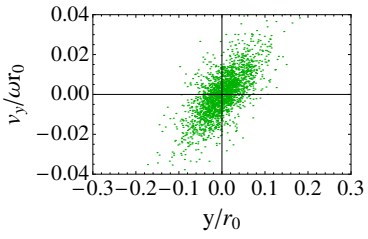
(f) $\omega t = -2/15\pi$ rad



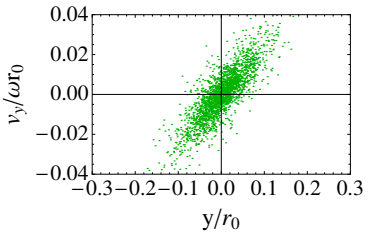
(g) $\omega t = 1/15\pi$ rad



(h) $\omega t = 4/15\pi$ rad

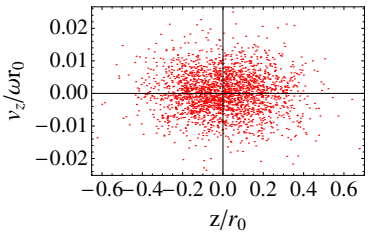


(i) $\omega t = 7/15\pi$ rad

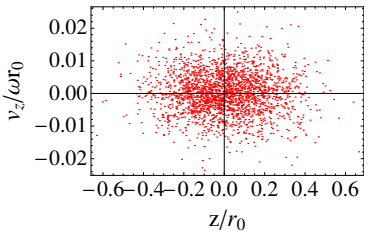


(j) $\omega t = 10/15\pi$ rad

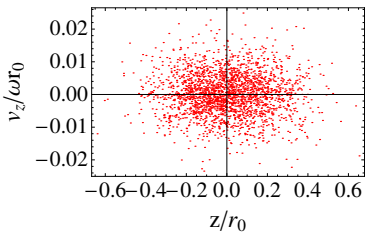
図 5.8: 冷却されたあとの位相空間分布 (y 方向)



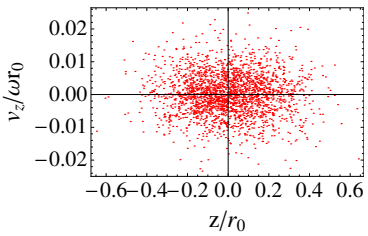
(a) $\omega t = -17/15\pi$ rad



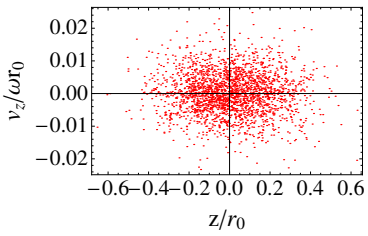
(b) $\omega t = -14/15\pi$ rad



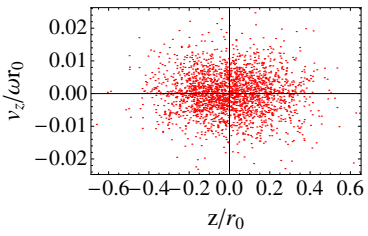
(c) $\omega t = -11/15\pi$ rad



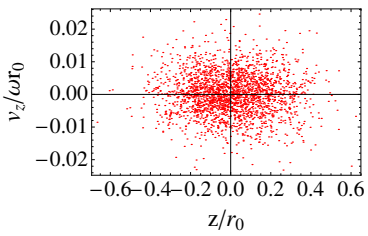
(d) $\omega t = -8/15\pi$ rad



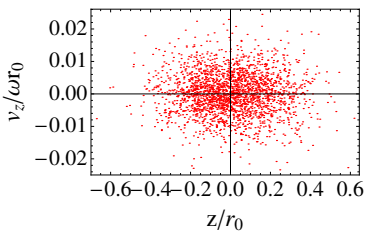
(e) $\omega t = -5/15\pi$ rad



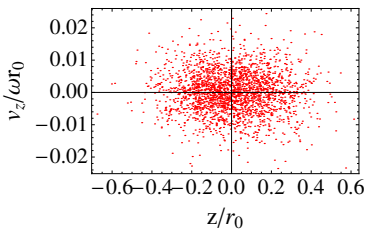
(f) $\omega t = -2/15\pi$ rad



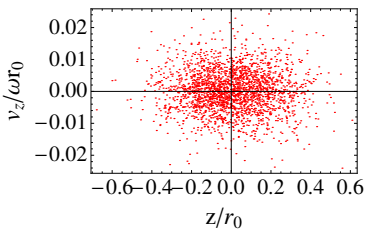
(g) $\omega t = 1/15\pi$ rad



(h) $\omega t = 4/15\pi$ rad



(i) $\omega t = 7/15\pi$ rad



(j) $\omega t = 10/15\pi$ rad

図 5.9: 冷却されたあとの位相空間分布 (z 方向)

第6章　リニアイオントラップの形状の最適化

2.3 で説明したように、ロッド電極形状に挿入する板状電極は、その厚みが十分に薄ければイオンの蓄積に理論的には影響を与えない。しかしながら、実際には板状電極は有限の厚みをもち、イオンの蓄積に対しなんらかの影響を与える可能性がある。この影響は非線形共鳴と呼ばれる現象と関係が深いと考えられる。

この章では、まずリニアイオントラップにおける非線形共鳴現象について説明する。次に、ロッド電極間に挿入された板状電極の影響について説明する。そして、ロッド電極間に板状電極を挿入したリニアイオントラップの形状の最適化について説明する。

6.1　リニアイオントラップでの非線形共鳴現象

リニアイオントラップにおける、非線形共鳴現象について概説する。一般的なリニアイオントラップ内部のポテンシャルは、式 (6.1) のように多重極展開で表現できる。

$$\phi(r,\theta,t) = (U + V cos\Omega t) \sum_{n=0}^{\infty} C_n \Big(\frac{r}{R}\Big)^n cosn\theta \tag{6.1}$$

ここで U, V はロッド電極に印加する電圧の直流成分と交流成分、Ω は角周波数、R はロッドの内接円半径であり、C_n が多重極展開のそれぞれの項の重みである。ロッド電極の断面形状が双曲線である場合、C_2 の項のみが現れる。C_2 に対応する項は、電位が式 (6.2) で与えられ、

$$\begin{aligned} \phi(r,\theta,t) &= \phi(x,y,t) \\ &= C_2(U + V cos\Omega t)\frac{x^2 - y^2}{R^2} \end{aligned} \tag{6.2}$$

その電場は式 (6.3) で与えられる。

$$\begin{aligned} E_x &= -2C_2(U + V cos\Omega t)\frac{x}{R^2} \\ E_y &= 2C_2(U + V cos\Omega t)\frac{y}{R^2} \end{aligned} \tag{6.3}$$

このように C_2 に対応する電場は座標に比例するので、しばしば線形項と呼ばれる。

ここで、ロッド電極の断面の形状が双曲線からズレた場合は、C_2 に加え $C_3,\ C_4,\ C_5, ...$ が現れる。C_2 以外に対応する項は、電場が座標に比例しないため、非線形項と呼ばれる。このとき、非線形項が形成する電場の影響により、イオン軌道が著しく不安定になる現象を非線形共鳴とよび、その発生条件は一般に式 (6.4) で与えられる [24]。

$$n_x\beta_x + n_y\beta_y = 2k \qquad (|n_x| + |n_y| = N) \tag{6.4}$$

ここで β_x, β_y は 2.2 で説明した、iso-β である。n_x, n_y, k は整数で、かつ $n_x n_y > 0$ であるときに式 (6.4) が満たされれば、非線形共鳴によりイオン軌道が不安定になる。図 6.1 には $k = 1$ で、$N = 3, 4, 5, 6$ の非線形共鳴が起きる条件 (非線形共鳴線) を描いた。とくに、k が小さい場合に、非線形共鳴の影響は強く現れる。また、非線形項のうちのある C_n の値が大きい場合、非線形共鳴の影響は $N = n, n-2, n-4, ..., 4, 3$ の非線形共鳴線で強く現れる。例えば、C_6 が大きい場合は、$N = 6, 4, 3$ の非線形共鳴線の影響が強くなる。また、ある N に対応するすべての非線形共鳴線は、安定領域の $a = 0$ 上の一点で交わる [23]。そして、この $a = 0$ 上の非線形共鳴線の交点は、N が大きくなるにつれ q 値の小さい方へ現れることが式 (6.4) から分かる。

ロッド電極間に板状電極を挿入したリニアイオントラップにおいても、挿入電極の影響で電場が乱れ、非線形共鳴現象が起こる可能性がある。

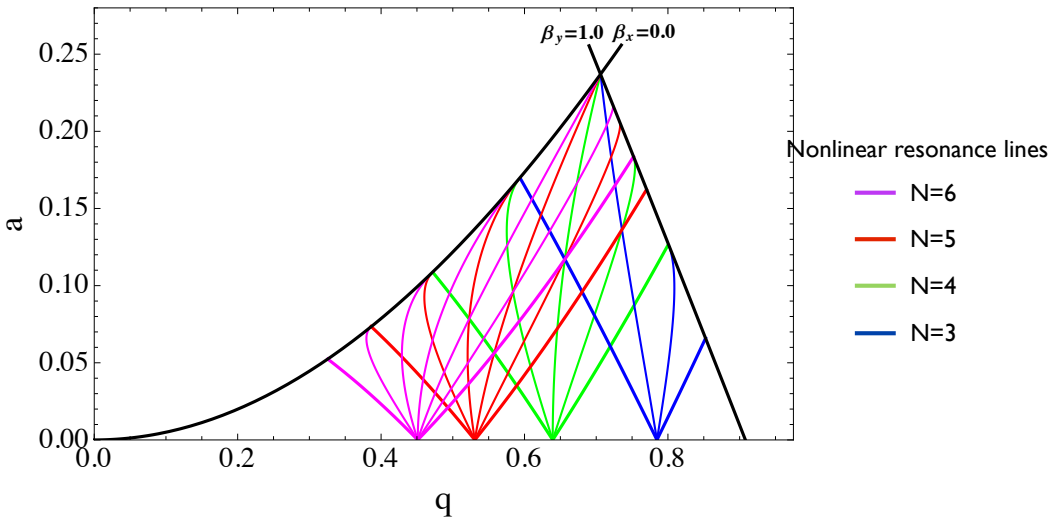


図 6.1: 電場が多重極展開で表現される場合の非線形共鳴が発生する条件

6.2 リニアイオントラップのロッド電極間に挿入された板状電極の影響

2.3 で説明したように、ロッド電極形状に挿入する板状電極は、その厚みが十分に薄ければイオンの蓄積に理論的には影響を与えない。しかしながら、実際には板状電極は有限の厚みをもち、イオンの蓄積に対しなんらかの影響を与える可能性がある。そこで、イオン軌道シミュレーションにより、ロッド電極間に挿入された板状電極がイオンの蓄積に及ぼす影響を定量的に評価することを試みた。

リニアイオントラップの中心付近から、イオンを 1000 個、適当な初期条件をもたせて運動させる。イオン軌道がロッド電極の内接円半径を超えた場合、そのイオンは蓄積できなかったと判断する。逆に一定時間以上、イオン軌道がロッド電極の内接円半径内部にあれば、そのイオンを蓄積できたと判断する。このとき、イオン蓄積効率を式 (6.5) で定義する。

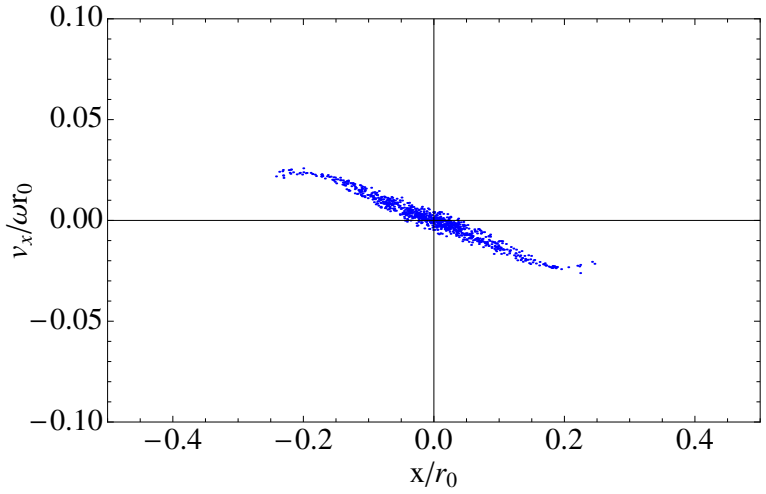
$$\text{トラップ効率} = \frac{\text{蓄積できたイオンの数}}{\text{最初のイオンの数}}[\%] \tag{6.5}$$

安定領域内の $a = 0$ 上の q 値を 200 分割し、トラップ効率を調べた。1000 個のイオンの運動の初期条件は、図 6.2 に位相空間分布で示す。図 6.2 において、r_0 はリニアイオントラップのロッドの内接円半径である。この初期条件は、5 で説明したようにイオンガイド 2 からリニアイオントラップにイオンを打ち込み、リニアイオントラップの中心に到達した時のイオンの x, y 方向の運動を調べたものであり、イオンと緩衝ガスの衝突は無視している。

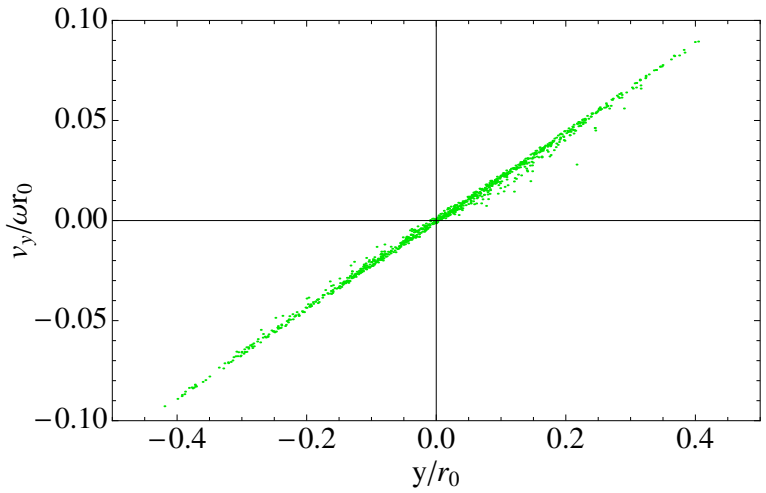
3 次元でこのようなシミュレーションを行うと膨大な時間がかかるので、図 6.3 に示したロッド電極と挿入電極 1,2 のみを考慮した 2 次元場を想定した。ロッドと挿入電極が形成する電場は、3.4 で説明したような、Fourier 展開による解析的な式を用いた。リニアイオントラップ内部の電場の Fourier 展開係数は、板状電極の厚み 1.0[mm], 2.0[mm] のそれぞれの場合について表 6.1,6.2 に示した。図 6.2 に示したようなイオンを打ち出し、0.3[ms] 飛行させて蓄積の成否を判断した。このシミュレーションを挿入電極がない場合、挿入電極の厚みが 1.0[mm], 2.0[mm] の場合について行った。

シミュレーション結果を図 6.4 に示す。$q = 0.0$〜0.55 でのトラップ効率は、挿入電極がない場合がもっと高く、挿入電極のが厚くなるにつれトラップ効率が低下している。$q = 0.55$ 以降では、挿入電極の厚みが 1.0[mm], 2.0[mm]

のどちらの場合でもトラップ効率が局所的に著しく低下する q 値がある。このように局所的にイオンのトラップ効率が低下する原因は、6.1 で説明した非線形共鳴現象によるものと考えられる。表 6.1, 6.2 を見ると、電場の Fourier 展開係数の C_2 以外のものでは C_6 が最大である。したがって、C_6 がトラップ効率低下と非線形共鳴の主因であると判断した。



(a) トラップ効率評価シミュレーションの 1000 個のイオンの x 方向の初期条件



(b) トラップ効率評価シミュレーションの 1000 個のイオンの y 方向の初期条件

図 6.2: トラップ効率評価シミュレーションの 1000 個のイオンの初期条件

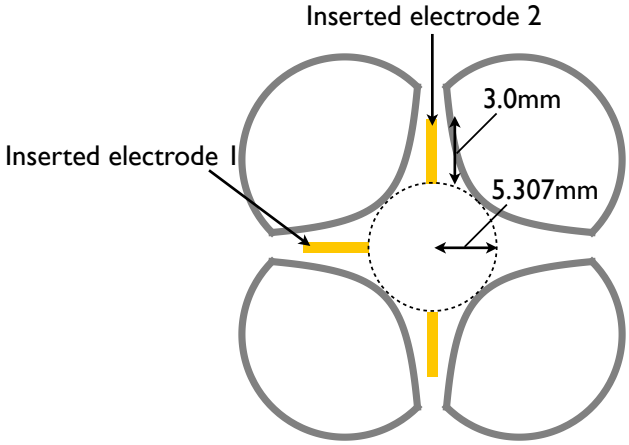


図 6.3: トラップ効率評価シミュレーションで考慮する電極

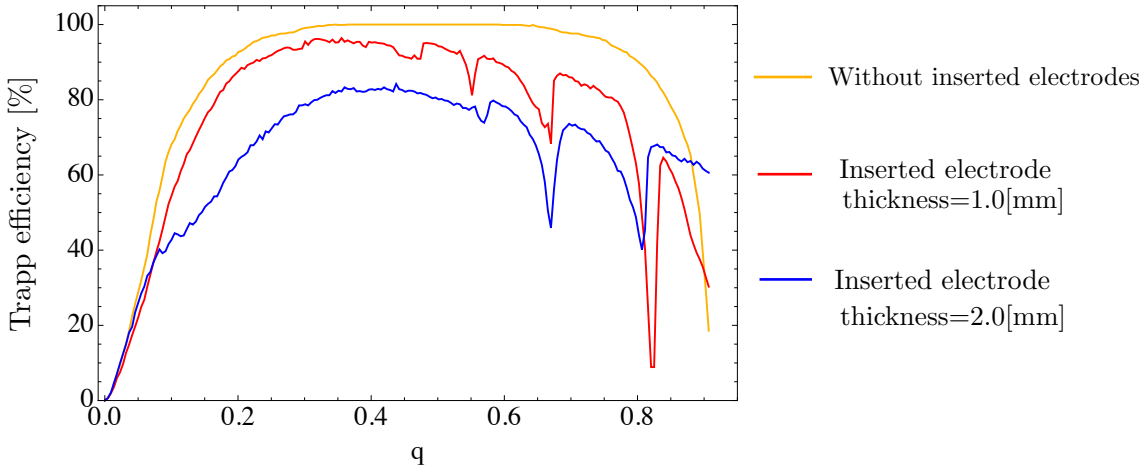


図 6.4: a=0 上でのトラップ効率

C_0=0.00000	C_1=0.01716	C_2=0.95702	C_3=0.02294
C_4=0.00000	C_5=−0.02491	C_6=0.10936	C_7=−0.02462
C_8=0.00000	C_9=0.02217	C_{10}=−0.08836	C_{11}=0.01960
C_{12}=0.00000	C_{13}=−0.01712	C_{14}=0.06823	C_{15}=−0.01508
C_{16}=0.00000	C_{17}=0.01317	C_{18}=−0.05214	C_{19}=0.01143
C_{20}=0.00000	C_{21}=0.00976	C_{22}=0.03807	C_{23}=−0.00819
D_0=0.00000	D_1=−0.01716	D_2=0.00000	D_3=0.02294
D_4=0.03428	D_5=0.02491	D_6=0.00000	D_7=−0.02462
D_8=−0.03327	D_9=−0.02217	D_{10}=0.00000	D_{11}=0.01960
D_{12}=0.02591	D_{13}=0.01712	D_{14}=0.00000	D_{15}=−0.01508
D_{16}=−0.01996	D_{17}=−0.01317	D_{18}=0.00000	D_{19}=0.01143
D_{20}=−0.01498	D_{21}=0.00976	D_{22}=0.00000	D_{23}=0.00819

表 6.1: 挿入電極が 1.0[mm] の場合の電場の Fourier 展開係数

C_0=0.00000	C_1=0.04916	C_2=0.80209	C_3=0.06148
C_4=0.00000	C_5=−0.06231	C_6=0.25761	C_7=−0.05573
C_8=0.00000	C_9=0.04345	C_{10}=−0.15677	C_{11}=0.03068
C_{12}=0.00000	C_{13}=−0.01956	C_{14}=0.06268	C_{15}=−0.01047
C_{16}=0.00000	C_{17}=0.00364	C_{18}=−0.02552	C_{19}=0.00156
C_{20}=0.00000	C_{21}=0.00486	C_{22}=0.01396	C_{23}=−0.00680
D_0=0.00000	D_1=−0.04916	D_2=0.00000	D_3=0.06148
D_4=0.08907	D_5=0.06231	D_6=0.00000	D_7=−0.05573
D_8=−0.07070	D_9=−0.04345	D_{10}=0.00000	D_{11}=0.03068
D_{12}=0.03521	D_{13}=0.01956	D_{14}=0.00000	D_{15}=−0.01047
D_{16}=−0.00967	D_{17}=−0.00364	D_{18}=0.00000	D_{19}=0.00156
D_{20}=−0.00482	D_{21}=0.00486	D_{22}=0.00000	D_{23}=0.00680

表 6.2: 挿入電極が 2.0[mm] の場合の電場の Fourier 展開係数

6.3　電極形状の最適化 1

6.2 では、ロッド電極間に板状電極を挿入したリニアイオントラップでは、電場の Fourier 展開係数の C_6 が大きくなることが原因となり、トラップ効率が低下する可能性を示した。そこで、トラップ効率改善のために、展開係数 C_6 を最小化するような電極形状・配置を模索した。

C_6 を最小化する方法として、リニアイオントラップのロッドの断面形状を変更する方法が考えられる。例えば、図 6.5 に示すようなロッドの断面形状が円であるリニアイオントラップが形成する電場の Fourier 展開係数をもとめると表 6.3 のようになる。この形状では Fourier 展開係数 C_6 は $C_6 < 0$ となっている。一方で、表 6.1,6.2 に示したように、ロッドの断面が双曲線状で挿入電極があるリニアイオントラップの場合は、$C_6 > 0$ となっている。これらの計算結果から、ロッド電極間に板状電極を挿入したリニアイオントラップのロッドの断面形状を円形 (扇形) に変更すれば、ロッドの Fourier 展開係数への負の寄与と、挿入電極の Fourier 展開係数への正の寄与が打ち消し合うような形状があると予想した。

そこで、ロッドの断面形状を扇形に変更したロッド電極間に板状電極を挿入したリニアイオントラップが形成する電場の Fourier 展開係数を計算し、Fourier 展開係数の C_6 が最小となる電極形状を探索した。円柱ロッドの角度 θ、半径 r をパラメーターとして、ロッドの内接円半径、挿入電極の位置・形状は定数とした。また、イオン排出時には挿入電極に高電圧が印加されるため、挿入電極とロッド電極に 1.0[mm] 以上の間隔が確保できる範囲内でパラメーターを調整した（図 6.6)。計算結果を表 6.4 に示す。$\theta = \pi/8$ において、r を変化させながら C_6 を計算したところ、$r = 17.55$[mm] と $r = 17.83$[mm] の間で、C_6 の値が正から負に変わることが分かった（表 6.4)。したがって、C_6 を最小化する r は $17.55 < r < 17.83$[mm] の範囲に存在すると判断した。2 分法によって $C_6 \to 0$ となる r の値を探索し、$\theta = \pi/8, r = 17.62$[mm] が C_6 を最小化するロッド形状と決定した。

C_6 を最小化する電極形状でのリニアイオントラップ内の電場の Fourier 展開係数を表 6.5 に示す。また、C_6 を最小化する電極形状で 6.2 と同様に、トラップ効率を評価するシミュレーションを行った。図 6.2 に示した初期条件のイオンを打ち出し、0.3[ms] 飛行させて、式 (6.5) で定義されるトラップ効率を評価した。トラップ効率のシミュレーション結果を図 6.7 に示す。C_6 を最小化

することにより、挿入電極のないロッドの断面形状が双曲線状のリニアイオントラップとほとんど遜色ないトラップ効率を達成できている。しかしながら、6.1 で説明した式 (6.4) で定義される $N = 3, 4$ の非線形共鳴の影響と思われる局所的なトラップ効率の低下が見られる。これは比較的に値が大きい Fourier 展開係数 C_{10}, C_{14} などの電場の影響であると考えられる。$C_6, C_{10}, C_{14}, \cdots$ の複数の Fourier 展開係数を同時に十分小さくするように電極形状を決定できれば、非線形共鳴を抑えることができる可能性がある。

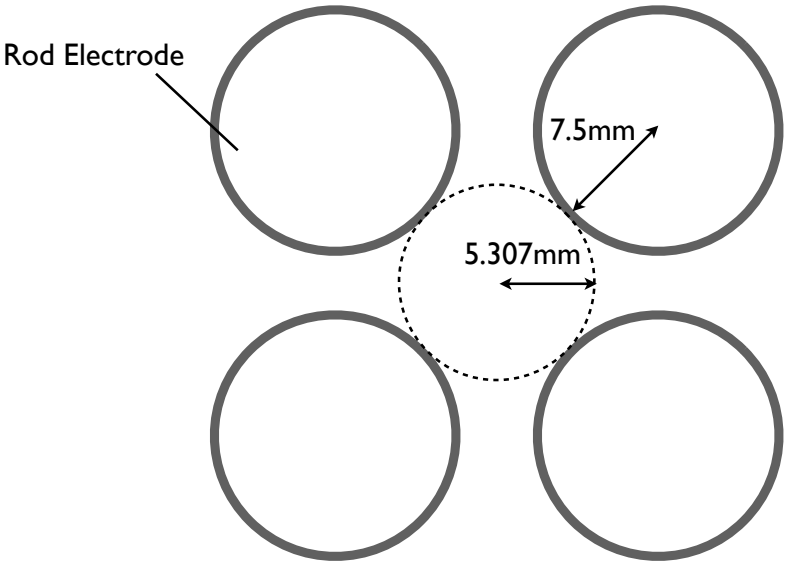


図 6.5: ロッド断面が円形のリニアイオントラップ

C_0=0.00000	C_1=0.00000	C_2=1.17180	C_3=0.00000
C_4=0.00000	C_5=0.00000	C_6=−0.01749	C_7=0.00000
C_8=0.00000	C_9=0.00000	C_{10}=−0.00256	C_{11}=0.00000
C_{12}=0.00000	C_{13}=0.00000	C_{14}=−0.00014	C_{15}=−0.00000
C_{16}=0.00000	C_{17}=0.00000	C_{18}=0.00000	C_{19}=0.00000
C_{20}=0.00000	C_{21}=0.00000	C_{22}=0.00000	C_{23}=0.00000
D_0=0.00000	D_1=0.00000	D_2=0.00000	D_3=0.00000
D_4=0.00000	D_5=0.00000	D_6=0.00000	D_7=0.00000
D_8=0.00000	D_9=0.00000	D_{10}=0.00000	D_{11}=0.00000
D_{12}=0.00000	D_{13}=0.00000	D_{14}=0.00000	D_{15}=−0.00000
D_{16}=0.00000	D_{17}=0.00000	D_{18}=0.00000	D_{19}=0.00000
D_{20}=0.00000	D_{21}=0.00000	D_{22}=0.00000	D_{23}=0.00000

表 6.3: 図 6.5 のロッド電極の断面形状が円状のリニアイオントラップの電場の Fourier 展開係数

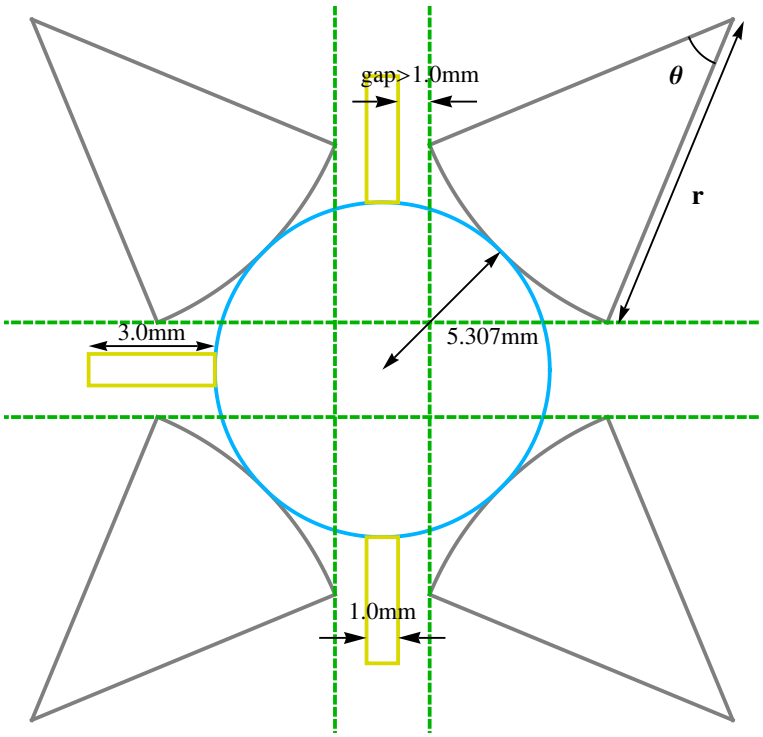


図 6.6: ロッドの断面形状を扇形に変更したロッド電極間に板状電極を挿入したリニアイオントラップ

θ	r[mm]	C_6	θ	r[mm]	C_6	θ	r[mm]	C_6
$\pi/2$	10.00	0.01492	$\pi/4$	10.00	0.01707	$\pi/8$	10.00	0.11016
	9.920	0.01534		10.01	0.01697		10.28	0.10390
	9.841	0.01578		10.03	0.01687		10.56	0.09775
	9.761	0.01622		10.04	0.01677		10.84	0.09170
	9.682	0.01666		10.05	0.01667		11.12	0.08579
	9.602	0.01711		10.07	0.01657		11.40	0.08003
	9.522	0.01757		10.08	0.01647		11.68	0.07443
	9.443	0.01803		10.09	0.01637		11.96	0.06899
	9.363	0.01850		10.11	0.01627		12.24	0.06373
	9.283	0.01897		10.12	0.01617		12.52	0.05866
	9.204	0.01945		10.14	0.01607		12.80	0.05379
	9.124	0.01994		10.15	0.01597		13.08	0.04910
	9.045	0.02043		10.16	0.01588		13.36	0.04462
	8.965	0.02093		10.18	0.01578		13.64	0.04034
	8.885	0.02144		10.19	0.01569		13.92	0.03627
	8.806	0.02195		10.20	0.01559		14.20	0.03240
	8.726	0.02247		10.22	0.01549		14.48	0.02873
	8.646	0.02300		10.23	0.01540		14.76	0.02525
	8.567	0.02354		10.24	0.01531		15.04	0.02198
	8.487	0.02408		10.26	0.01521		15.32	0.01889
	8.408	0.02463		10.27	0.01512		15.60	0.01600
	8.328	0.02519		10.28	0.01503		15.88	0.01328
	8.248	0.02575		10.30	0.01493		16.15	0.01083
	8.169	0.02633		10.31	0.01484		16.43	0.00845
	8.089	0.02691		10.32	0.01475		16.71	0.00623
	8.009	0.02750		10.34	0.01466		16.99	0.00417
	7.930	0.02810		10.35	0.01457		17.27	0.00226
	7.850	0.02871		10.36	0.01448		17.55	0.00048
	7.771	0.02933		10.38	0.01439		17.83	-0.00116
	7.691	0.02996		10.39	0.01430		18.11	-0.00268

表 6.4: 電極形状と展開係数 C_6 の関係

C_0=0.00000	C_1=0.00327	C_2=1.02700	C_3=0.00768
C_4=0.00000	C_5=−0.01036	C_6=0.00006	C_7=−0.01115
C_8=0.00000	C_9=0.01034	C_{10}=−0.04770	C_{11}=0.00904
C_{12}=0.00000	C_{13}=−0.00760	C_{14}=0.03298	C_{15}=−0.00638
C_{16}=0.00000	C_{17}=0.00531	C_{18}=−0.02131	C_{19}=0.00442
C_{20}=0.00000	C_{21}=−0.00365	C_{22}=0.01396	C_{23}=−0.00230
D_0=0.00000	D_1=−0.00327	D_2=0.00000	D_3=0.00768
D_4=0.01309	D_5=0.01036	D_6=0.00000	D_7=−0.01115
D_8=−0.01536	D_9=−0.01034	D_{10}=0.00000	D_{11}=0.00904
D_{12}=0.01174	D_{13}=0.00760	D_{14}=0.00000	D_{15}=−0.00638
D_{16}=−0.00822	D_{17}=−0.00531	D_{18}=0.00000	D_{19}=0.00442
D_{20}=0.00568	D_{21}=0.00365	D_{22}=0.00000	D_{23}=−0.00296

表 6.5: C_6 を最小化した電極形状での電場の Fourier 展開係数

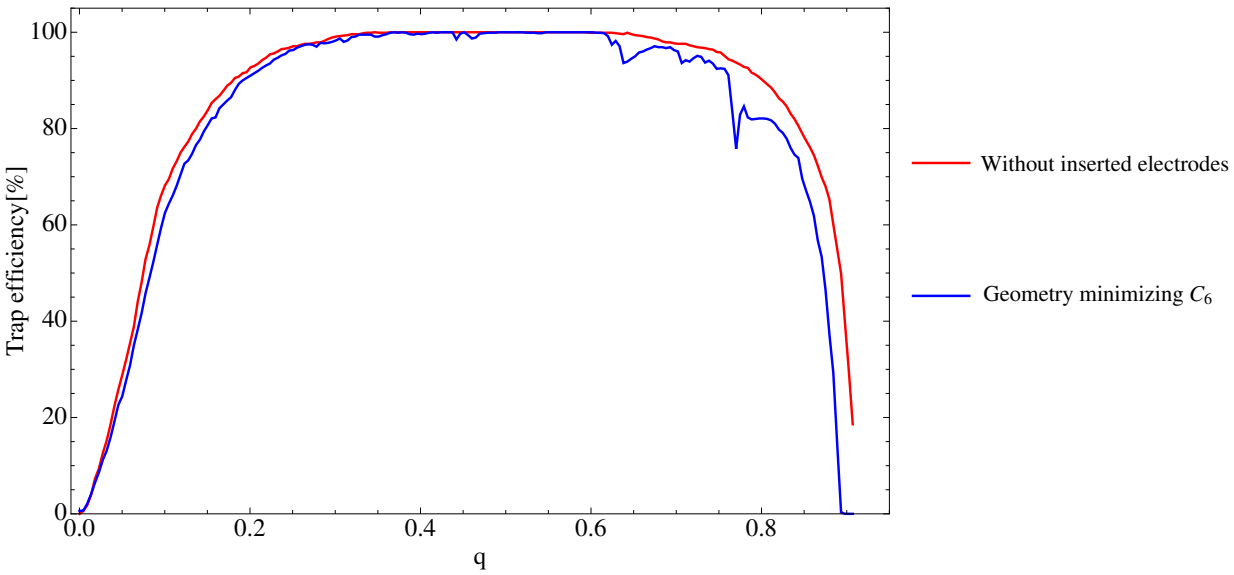


図 6.7: C_6 を最小化する電極形状でのトラップ効率

6.4　電極形状の最適化２

次に挿入電極 3 を含めたリニアイオントラップの最適な形状を検討した。6.3 で述べたように、ロッド電極の断面形状を扇形に変更することにより、ロッド電極間に板状電極を挿入したリニアイオントラップの電場の Fourier 展開係数 C_6 を最小化し、6.2 で定義したトラップ効率を向上させることができた。そこで、ロッド電極と挿入電極 1,2 は 6.3 でもとめた形状・配置とし、挿入電極 3 は C_6 への影響を無視できるような位置に配置することを考えた。

まず、Fourier 展開係数の C_6 がどの程度小さければ 6.2 で定義したトラップ効率への影響を無視できるかを定量的に見積もった。図 6.1 に示されるように、C_6 の影響が顕著に表れる非線形共鳴の条件は、$a = 0$ 上では $q = 0.451,\ 0.64,\ 0.785$ である。このような非線形共鳴が顕著に表れる点においても、トラップ効率が低下しないような C_6 の上限値を見積もる。$C_2 = 1.0$ で C_6 を変化させながらトラップ効率をシミュレーションした結果を図 6.8 に示す。ここで電場はトラップ効率に対する C_6 の影響だけを見積もるために、C_2, C_6 のみ考慮している。$q = 0.451,\ 0.64,\ 0.785$ それぞれの点で、トラップ効率は $C_6 \leq 10^{-3}$ 程度以下になればトラップ効率は低下しないという結果を得た。

そこで、挿入電極 3 は $C_6 \leq 10^{-3}$ となる範囲で配置することにした。図 6.10 はトラップ軸から挿入電極 3 までの距離 L をパラメータとし (図 6.9)、C_6 の値を計算した結果である。これらの計算結果から、挿入電極 3 はトラップ軸から 7.0[mm] 以上、離して配置するのが妥当であると判断した。

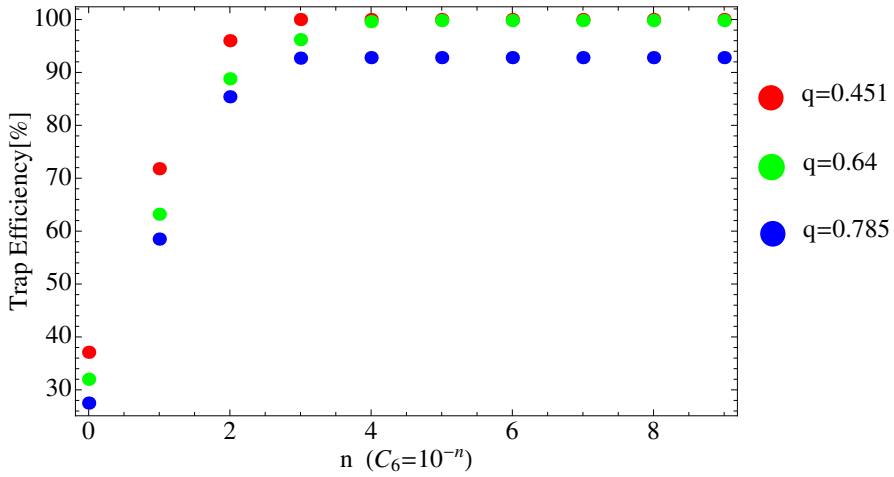


図 6.8: C_6 のトラップ効率に及ぼす影響

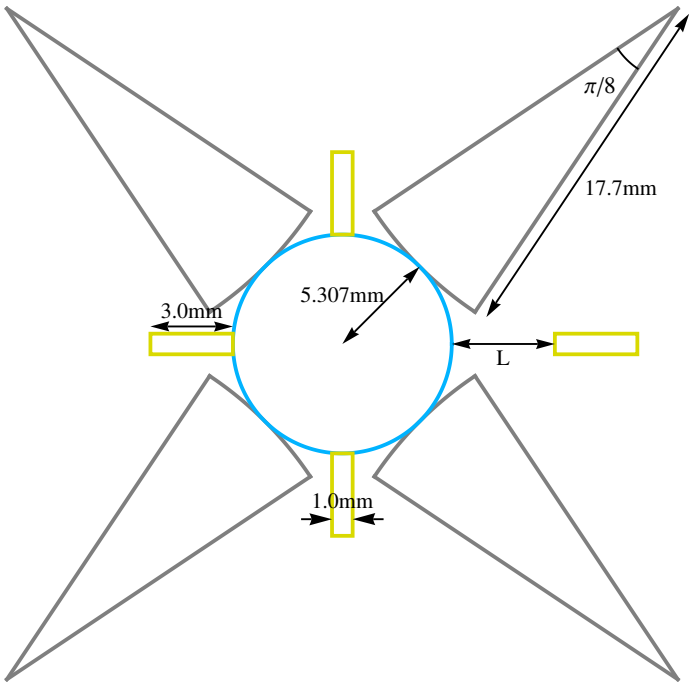


図 6.9: 挿入電極３を考慮して C_6 を計算

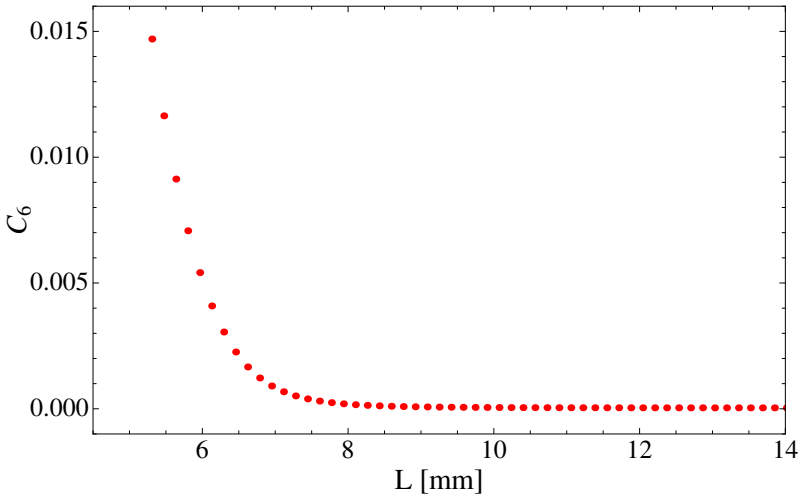


図 6.10: 挿入電極 3 を考慮して C_6 を計算した結果

6.5　組み立て誤差のトラップへの影響

6.3, 6.4 では挿入電極のトラップ効率への悪影響を打ち消すようなトラップの形状・配置を示したが、現実には、電極の加工誤差・組立て誤差があるため、理想的な形状・配置からズレてしまう。ここでは、各電極の位置がズレた場合にどの程度の電場に影響を与えるか計算し、トラップ効率の低下を防ぐにはどの程度の組み立て精度が求められるかを見積もった。ロッド電極と挿入電極の各々が (i)〜(iv) のような組み立てのズレをもちうるが、ロッド電極は理想的な位置に固定し、挿入電極の位置がズレる場合の組み立て誤差の影響を見積もった。

(i) 電極とトラップ軸との距離がズレる (図 6.11)

(ii) 電極が横方向にズレる (図 6.7)

(iii) 電極の角度がズレる (図 6.8)

(iv)i〜iii の複合

組立て誤差の影響は、Fourier 展開係数の C_6 を指標とした。計算結果を表 6.6〜6.8 に示す。表 6.6 から、各々の挿入電極のトラップ軸からの距離が最大で ± 0.2[mm] ズレる場合、C_6 は最大で $\sim 10^{-2}$ 程度である。表 6.7 から、各々の挿入電極が横方向に最大で ± 0.2[mm] ズレる場合、C_6 は最大で $\sim 10^{-3}$ 程度である。表 6.8 から、各々の挿入電極の角度が最大で ± 18[°] ズレる場合、C_6 は最大で $\sim 10^{-4}$ 程度である。これらの計算結果から、挿入電極のトラップ軸からの距離がズレる場合が、もっとも C_6 に与える影響が大きい。したがって、挿入電極のトラップ軸からの距離が 10^{-1}[mm] 程度の組み立て精度を達成できるように設計を行うべきである。

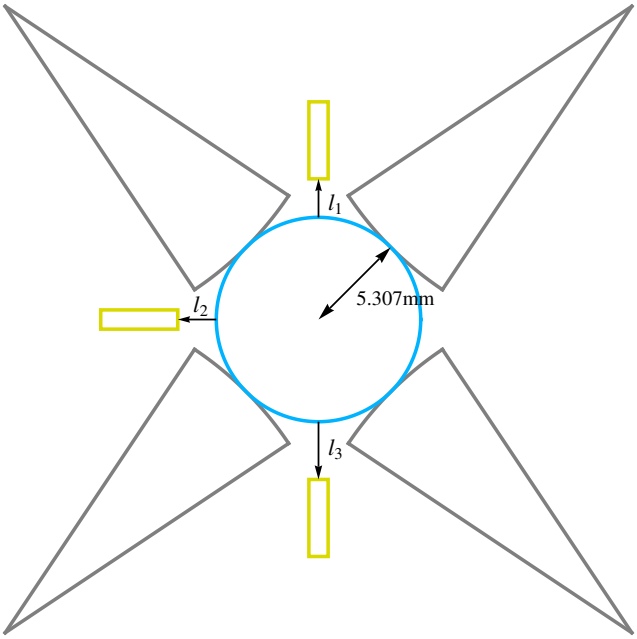


図 6.11: 挿入電極のトラップ軸からの距離がズレている場合 (l は内接円をの位置を基準とし、トラップ軸から離れる方向を正とする)

$l_1[mm]$	$l_2[mm]$	$l_3[mm]$	C_6
0.2000	0.2000	0.2000	-0.01092
0.2000	0.2000	0.0000	-0.007287
0.2000	0.2000	-0.2000	-0.002710
0.2000	0.0000	0.2000	-0.007289
0.2000	0.0000	0.0000	-0.003661
0.2000	0.0000	-0.2000	0.0009163
0.2000	-0.2000	0.2000	-0.002713
0.2000	-0.2000	0.0000	0.0009144
0.2000	-0.2000	-0.2000	0.005490
0.0000	0.2000	0.2000	-0.007287
0.0000	0.2000	0.0000	-0.003659
0.0000	0.2000	-0.2000	0.0009188
0.0000	0.0000	0.2000	-0.003661
0.0000	0.0000	0.0000	-0.00003260
0.0000	0.0000	-0.2000	0.004544
0.0000	-0.2000	0.2000	0.0009144
0.0000	-0.2000	0.0000	0.004542
0.0000	-0.2000	-0.2000	0.009118
-0.2000	0.2000	0.2000	-0.002710
-0.2000	0.2000	0.0000	0.0009188
-0.2000	0.2000	-0.2000	0.005497
-0.2000	0.0000	0.2000	0.0009163
-0.2000	0.0000	0.0000	0.004544
-0.2000	0.0000	-0.2000	0.009121
-0.2000	-0.2000	0.2000	0.005490
-0.2000	-0.2000	0.0000	0.009118
-0.2000	-0.2000	-0.2000	0.01369

表 6.6: 挿入電極とトラップ軸との距離がズレた場合の展開係数 C_6(l は内接円をの位置を基準とし、トラップ軸から離れる方向を正とする)

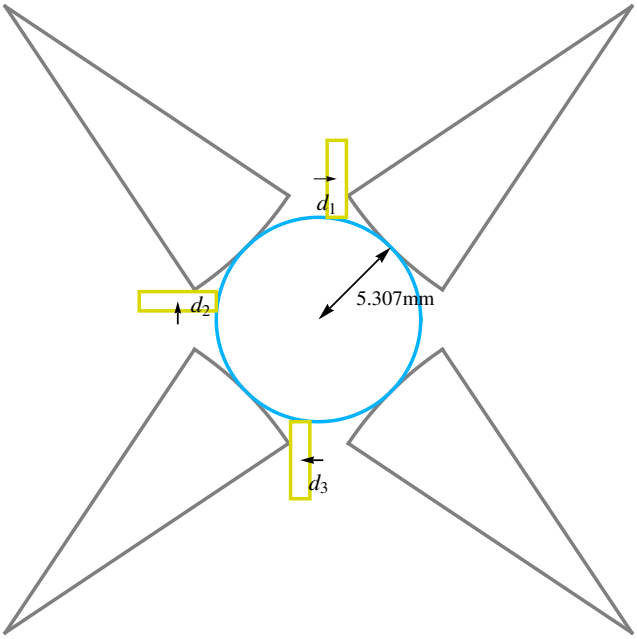


図 6.12: 挿入電極が横にズレている場合 (d はトラップ軸を中心として、時計回り方向を正とする)

$d_1[mm]$	$d_2[mm]$	$d_3[mm]$	C_6
-0.2000	-0.2000	-0.2000	0.001534
-0.2000	-0.2000	0.0000	0.001002
-0.2000	-0.2000	0.2000	0.001490
-0.2000	0.0000	-0.2000	0.0009754
-0.2000	0.0000	0.0000	0.0004680
-0.2000	0.0000	0.2000	0.0009803
-0.2000	0.2000	-0.2000	0.001436
-0.2000	0.2000	0.0000	0.0009528
-0.2000	0.2000	0.2000	0.001490
0.0000	-0.2000	-0.2000	0.001021
0.0000	-0.2000	0.0000	0.0004772
0.0000	-0.2000	0.2000	0.0009528
0.0000	0.0000	-0.2000	0.0004865
0.0000	0.0000	0.0000	-0.00003260
0.0000	0.0000	0.2000	0.0004680
0.0000	0.2000	-0.2000	0.0009725
0.0000	0.2000	0.0000	0.0004772
0.0000	0.2000	0.2000	0.001002
0.2000	-0.2000	-0.2000	0.001528
0.2000	-0.2000	0.0000	0.0009725
0.2000	-0.2000	0.2000	0.001436
0.2000	0.0000	-0.2000	0.001017
0.2000	0.0000	0.0000	0.0004865
0.2000	0.0000	0.2000	0.0009754
0.2000	0.2000	-0.2000	0.001528
0.2000	0.2000	0.0000	0.001021
0.2000	0.2000	0.2000	0.001534

表 6.7: 挿入電極が横にズレている場合の展開係数 C_6(d はトラップ軸を中心として、時計回り方向を正とする)

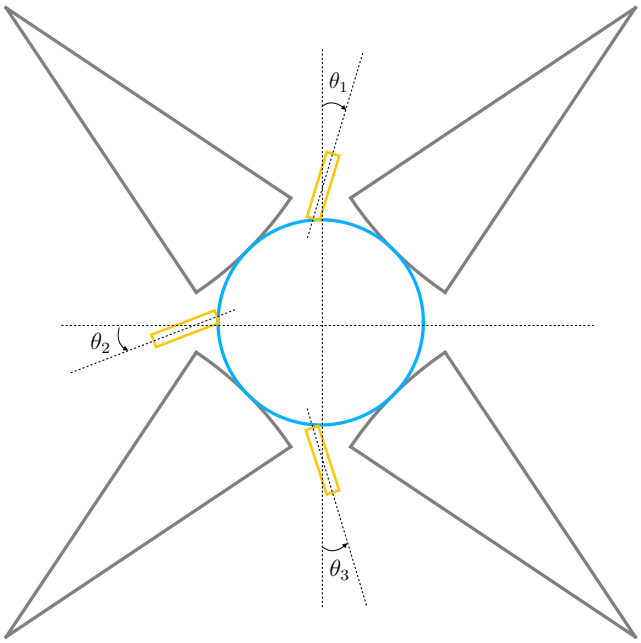


図 6.13: 挿入電極の角度がズレている場合（θ は時計回りを正とする）

$\theta_1[^\circ]$	$\theta_2[^\circ]$	$\theta_3[^\circ]$	C_6
-18.00	-18.00	-18.00	0.0003674
-18.00	-18.00	0.00	0.0002435
-18.00	-18.00	18.00	0.0003442
-18.00	0.00	-18.00	0.0001736
-18.00	0.00	0.00	0.00009014
-18.00	0.00	18.00	0.0002323
-18.00	18.00	-18.00	0.0002059
-18.00	18.00	0.00	0.0001617
-18.00	18.00	18.00	0.0003442
0.00	-18.00	-18.00	0.0002237
0.00	-18.00	0.00	0.00008041
0.00	-18.00	18.00	0.0001617
0.00	0.00	-18.00	0.00007031
0.00	0.00	0.00	-0.00003260
0.00	0.00	18.00	0.00009014
0.00	18.00	-18.00	0.0001442
0.00	18.00	0.00	0.00008041
0.00	18.00	18.00	0.0002435
18.00	-18.00	-18.00	0.0003069
18.00	-18.00	0.00	0.0001442
18.00	-18.00	18.00	0.0002059
18.00	0.00	-18.00	0.0001926
18.00	0.00	0.00	0.00007031
18.00	0.00	18.00	0.0001736
18.00	18.00	-18.00	0.0003069
18.00	18.00	0.00	0.0002237
18.00	18.00	18.00	0.0003674

表 6.8: 挿入電極の角度がズレている場合の展開係数 C_6（θ は時計回りを正とする）

第7章　リニアイオントラップからのイオン排出シミュレーション

2.1 で述べたように、ロッド電極間に板状電極を挿入したリニアイオントラップに蓄積されたイオンを MULTUM-SⅡ の入射電極の位置で空間的に収束させ、検出器の位置で時間的に収束させて排出することが望ましい。そこで、6 で考案した、6.2 で定義したトラップ効率が最大となる、ロッドの電極形状を扇型に変更したロッド電極間に板状電極を挿入したリニアイオントラップからイオンを排出するシミュレーションを作成した。5.2 で求めた、イオントラップ内部に蓄積されたイオンを任意の位置で時間的・空間的に収束させて排出する各電極への電圧の印加方法について調べることを目標とした。

イオン排出シミュレーションを全て３次元の表面電荷法プログラムを用いて計算した場合、電極の数が多いため計算に時間がかかりすぎてしまう。計算量を軽減するために排出シミュレーションを、リニアイオントラップ内部から引き出し電極１までの前半と、引き出し電極１から引き出し電極２までの後半に分けた。排出シミュレーションのメッシュを図 7.1 に示す。前半では３次元の表面電荷法によりイオン軌道を計算し、後半は電極形状が回転対称なものを想定したので回転対称場の表面電荷法を用いている。このようにすることで、電極が形成する電場の計算を高速化させた。

ここでは、引き出し電極 2 を通過したところでイオンを時間的・空間的に収束させて排出する方法について調べることにした。電極が形成する電場を高速に計算する工夫を取り入れたが、それでも 5.2 で求めた、イオントラップ内部に蓄積されたイオン (7110 個) をすべて排出するシミュレーションを行うと長時間かかるため、電極に印加する電圧を変えて最適な値をもとめるには不利である。そこで、5.2 で求めたイオントラップ内部に蓄積されたイオンの空間分布を参考にして、リニアイオントラップ内部に 5 個のイオンを空

間的に分布させて配置し、まずその 5 個のイオンを時間的・空間的に収束させて排出する電圧条件を調べた。

5 個のイオンは、リニアイオントラップの中心から x 方向に ±0.5[mm]、y 方向に ±0.5[mm]、z 方向に ±1.5[mm] ずつ等間隔にずらして配置した。この 5 個のイオンの初期の運動状態は静止した状態にした。このような状況で図 2.13 に示した各電極の印加電圧を変化させて時間収束性・空間収束性を調べたところ、挿入電極 1：1000[V]、挿入電極 2：700[V]、挿入電極 3：0[V]、引き出し電極 1：0[V]、引き出し電極 2：−700[V] の条件のとき、引き出し電極 2 を通過した位置で 5 個のイオンの z 方向の広がりが 0.44[mm]、y 方向の広がりが 1.27[mm]、飛行時間の差が 9[ns] 以内となり、本研究で探索した中で最も時間収束性と空間収束性の良い電圧条件であった。この電圧条件での 5 個のイオンの軌道を図 7.2 に示した。

先に述べた電圧条件で、5.2 で求めたイオントラップ内部に蓄積された 7110 個のイオンを排出するシミュレーションを行った。シミュレーション結果を図 7.3,7.4 に示す。静止した 5 個のイオンを排出した場合は飛行時間の差が 9[ns] 以内に収まっていたが、5.2 で求めたイオントラップ内部に蓄積されたイオンを排出した場合は図 7.3 に示されるように 20[ns] 程度の半値幅をもっている。また、静止した 5 個のイオンを排出した場合は引き出し電極 2 を通過した位置での z 方向の広がりが 0.44[mm]、y 方向の広がりが 1.27[mm] となっていたが、5.2 で求めたイオントラップ内部に蓄積されたイオンを排出した場合は図 7.4 に示したように z 方向の広がりが 1.0[mm]、y 方向の広がりが 3.0[mm] 程度になった (全イオンの約 80 %が含まれる範囲)。

このように時間収束性・空間収束性ともに悪化した原因として、配置した 5 個のイオンに速度分布を持たせずに電圧条件を探索したことが考えられる。改善策として、5.2 で求めたイオントラップ内部に蓄積されたイオンの空間分布・運動状態を反映したイオンを数個リニアイオントラップの内部に配置し、それらのイオンを時間的・空間的に収束させて排出する電圧条件を見つける方法が考えられる。また、イオン排出シミュレーションを simplex 法などの最適値を探索するアルゴリズムと組み合わせることで、最適な電圧条件を定量的に決定することが望ましいと考えられる。

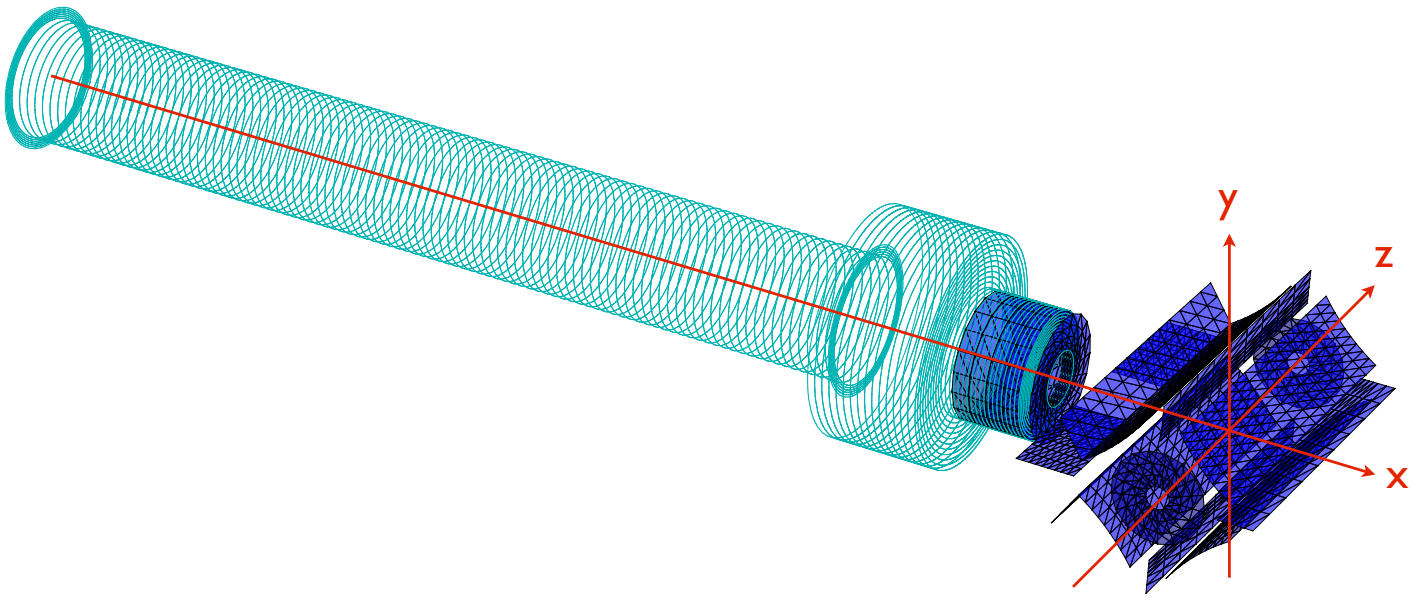
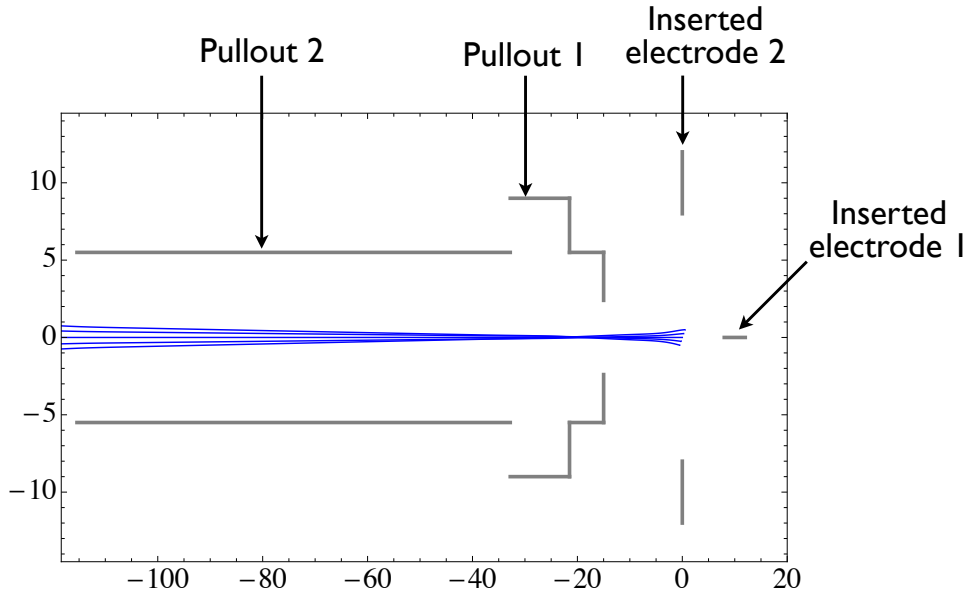
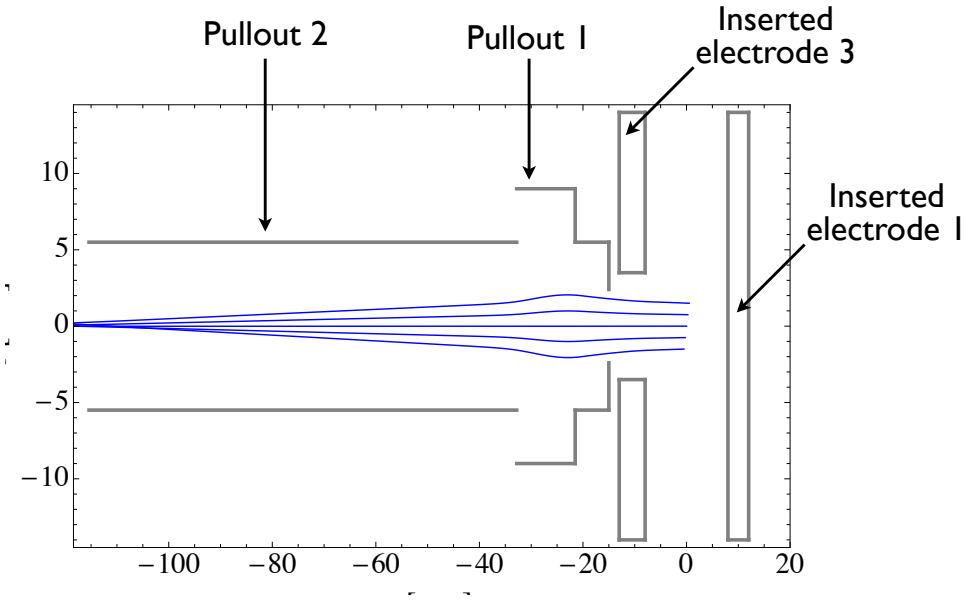


図 7.1: イオン排出シミュレーションのメッシュ



(a) イオン排出時の x-y 方向のイオン軌道



(b) イオン排出時の x-z 方向のイオン軌道

図 7.2: イオン排出時のイオン軌道

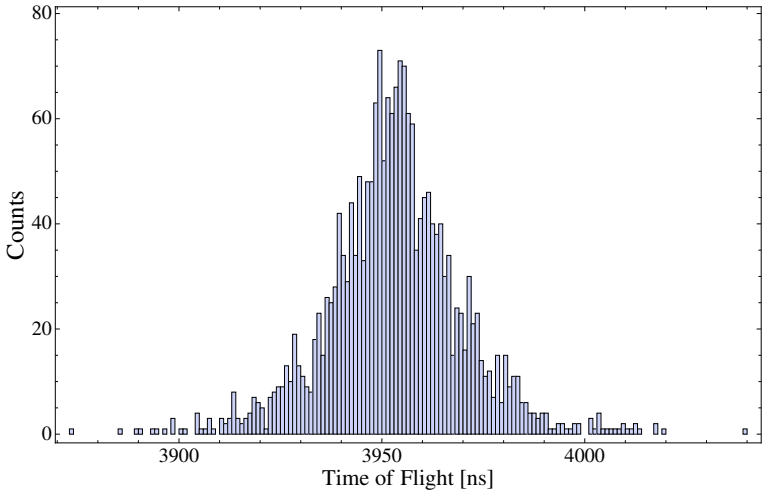


図 7.3: 引き出し電極２の出口での飛行時間分布

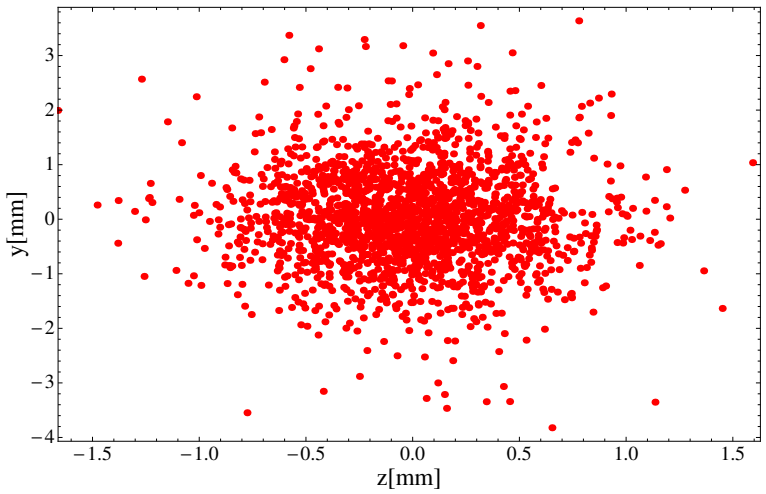


図 7.4: 引き出し電極２の出口での空間分布

第8章　まとめ

8.1　まとめ

本研究では、大気圧イオン源と MULTUM-S の接続インターフェースとなる、ロッド電極間に板状電極を挿入したリニアイオントラップ装置の性能向上を目的として、当装置でのイオン軌道シミュレーションを行った。まず、イオンと中性ガスとの衝突現象を考慮した表面電荷法プログラムを作成し、イオンガイドからリニアイオントラップに導入されたイオンの、トラップ内部での運動状態と空間分布について調べた。

また、二次元のイオン軌道シミュレーションプログラムを作成し、ロッド電極間に板状電極を挿入したリニアイオントラップのトラップ効率を評価し、挿入電極の影響により電場が乱れ、トラップ効率が低下することを示した。さらに、ロッド断面の形状を双曲線状から扇状に変更することで、イオンの蓄積効率を向上する可能性があることを示した。装置の組み立て誤差がトラップ効率に与える影響についても系統的に評価し、挿入電極とトラップ軸間の距離のズレが最もトラップ効率を低下させることを示し、装置製作時に要求される組み立て精度を見積もった。

そして、断面形状が扇型のロッド電極間に板状電極を挿入したリニアイオントラップからのイオン排出過程をシミュレーションし、各電極の役割と印加する適当な電圧値について定性的に調べた。イオン排出過程のシミュレーションについては、simplex 法などの最適値探索アルゴリズムと組み合わせることで最適な電圧条件を求め、イオン排出時の時間収束性・空間収束性について定量的に評価することが望ましい。

第9章　付録

本研究のシミュレーションで使用した, 表面電荷法の C++プログラムとその使用方法を示す．３次元場・２次元場・回転対称場のいずれのプログラムも, 仮想電荷の電荷量を計算するまでの部分と, 電場を計算する部分を別々のプログラムに分けている。

9.1　２次元場の表面電荷法プログラム

仮想電荷（線電荷）の線電荷密度を計算する C++プログラム imaginary_2D.cpp をソースコード 9.3 に示す。また、電場を計算する C++プログラム field_2D.cpp をソースコード 9.4 に示す。

imaginary_2D.cpp は電極の形状と境界条件を指定すれば、仮想電荷の量を計算する. 電極形状の指定は入力ファイル node.txt と element.txt で行う。境界条件の指定は入力ファイル boundary_condition.txt で行う。これらの入力ファイルの作成の大半は Mathematica7.0.0 を用いて行った。node.txt は節点の座標の指定と番号付けを担い、図 9.1 のような２列の数値が並んだデータ形式である。１列目, ２列目の数値はそれぞれ節点の x,y 座標を表し、n 行目のデータは第 n 番の節点に対応する。すべての節点は互いに重複していはならない。element.txt は要素がどの節点で構成されるかを表し、図 9.2 のような２列の正数が並んだデータ形式である。１列目, ２列目の正数はそれぞれ節点の番号を表し、ひとつの行である要素を構成する節点の組み合わせを表す。boundary_condition.txt は境界条件となる節点の電圧を表し、図 9.3 のような１列の数値が並んだデータ形式である。n 行目の数値は第 n 番の節点の電圧を表す。このような入力ファイルを作成し、imaginary_2D.cpp を実行すると、出力ファイル charge_density.txt が出力される。charge_density.txt は図 9.4 に示したような１列の数値が並んだデータ形式であり、仮想電荷（線電荷）の線電荷密度を表す。

そして、field_2D.cpp を実行すれば、charge_density.txt と caluculation_point.txt

を読み込んで 2 次元場を計算する。計算結果はと field.txt して出力される。caluculation_ point.txt は図 9.5 のような 2 列の数値データで、 1 列目, 2 列目の数値はそれぞれ電場を計算する点の x,y 座標を表している。field.txt は図 9.6 のような 2 列の数値データで、 1 列目, 2 列目の数値はそれぞれ x 方向の電場,y 方向の電場を表している。

ソースコード 9.1: imaginary_2D.cpp（仮想電荷の電荷量を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //置換定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 1500//要素数
9  #define N_node 1500//節点数
10 #define N_image NG*N_ele//仮想電荷の数
11 #define EPS 1.0e-5//ガウス消去法の許容誤差
12
13 //グローバル変数
14 int element[N_ele][2];//要素の情報を格納
15 double node[N_node][2];//節点の情報を格納
16 double x[NG],w[NG];//積分点とその重みを格納
17 double charge_density[N_image];//仮想電荷（線電荷）の線電荷密度を格納
18
19 //関数のプロトタイプ宣言
20 int get_element();//要素の情報を得る
21 int get_node();//節点の情報を得る
22 double get_boundary(double V_node[]);//境界条件の情報を得る
23 void gauss(int m);//ガウス積分公式の積分点、重みを決める
24 double cal_Aij1(int i, int k);//電位係数の計算 1
25 double cal_Aij2(int i, int k);//電位係数の計算 2
26 void cal_imaginary(double charge_node[]);//仮想電荷の電荷量を計算する
27
28 //ガウス消去法に用いる関数のプロトタイプ宣言
29 void DATAIN(const char name[], int nr, int nc,double ma[], double Aij[][N_node]);//電位
       係数の行列を作成
30 void DATAIN2(const char name[], int nr, int nc, double ma[], double V_node[]);//境界条件
       の行列を作成
31 int PIVOT(int *num, int nr, int k, double ma[]);//ピボットの選択
32 int GAUSS(int nr, double ma[], double mb[], double mx[]);//ガウス消去法
33
34 //メイン関数
35 int main()
36 {
37         //要素、節点の情報を得る
38         get_element();
39         get_node();
40
41         //ガウス積分の積分点と重みを決める
42         gauss(NG);
```

```
43
44          //節点の電位係数の計算
45          static double Aij[N_node][N_node]={0.0};
46          for (int k=0; k<N_ele ; k++)
47          {
48                  for (int i=0; i<N_node; i++)
49                  {
50                          int k1,k2;
51                          k1=element[k][0]-1; k2=element[k][1]-1;
52
53                          Aij[i][k1] += cal_Aij1(i,k);
54                          Aij[i][k2] += cal_Aij2(i,k);
55                  }
56          }
57
58          //節点の表面電荷密度をもとめる
59          int nr=N_node,nc;
60          int flag;
61          static double ma[N_node*N_node];
62          double mb[N_node],charge_node[N_node];
63          nc=nr;
64          DATAIN("Aij",nr,nc,ma,Aij);
65          double V_node[N_node];
66          get_boundary(V_node);
67          DATAIN2("V_node",nr,1,mb,V_node);
68          flag=GAUSS(nr,ma,mb,charge_node);
69          if(flag) printf("\n 計算不能\n");
70
71          //仮想電荷（線電荷）の線電荷密度を計算する
72          cal_imaginary(charge_node);
73
74          //仮想電荷の電荷量をテキストデータに出力
75          FILE *fp;
76          fp = fopen( "charge_density.txt", "w" );
77          if( fp == NULL )
78          {
79                  puts( "ファイルが開けません\n" );
80                  return 1;
81          }
82          for(int j=0; j<N_image; ++j)
83          {
84                  fprintf( fp, "%16.16lf", charge_density[j]);
85                  fprintf( fp, "\n" );
86          }
87          fclose( fp );
```

```
88          return 0;
89  }
90
91  //関数の定義
92  int get_node()
93  {
94          FILE *fp;
95          char lBuf[512],*p;
96          if ((fp = fopen("node.txt","r")) == NULL){
97                  fprintf(stderr,"ファイルが開けません\n");
98                  return 1;
99          }
100         for(int m=0; m<N_node; m++ ) {
101                 fgets( lBuf, sizeof( lBuf ), fp );
102                 p = strtok( lBuf, "␣␣␣␣␣" );
103                 for (int n=0; n<2 && p!=NULL; n++) {
104                         sscanf( p, "%lf", &node[m][n] );
105                         p = strtok( NULL, "␣␣␣␣␣" );
106                 }
107         }
108         fclose(fp);
109         return 0;
110 }
111
112 int get_element()
113 {
114         FILE *fp;
115         char lBuf[512],*p;
116         if ((fp = fopen("element.txt","r")) == NULL){
117                 fprintf(stderr,"ファイルが開けません\n");
118                 return 1;
119         }
120         for(int m=0; m<N_ele; m++ ) {
121                 fgets( lBuf, sizeof( lBuf ), fp );
122                 p = strtok( lBuf, "␣␣␣␣␣" );
123                 for (int n=0; n<2 && p!=NULL; n++) {
124                         sscanf( p, "%d", &element[m][n] );
125                         p = strtok( NULL, "␣␣␣␣␣" );
126                 }
127         }
128         fclose(fp);
129         return 0;
130 }
131
132 void gauss(int m)
```

```
133 {
134         if (m==2) {
135                 x[0]=−0.5773502691896257;
136                 x[1]=0.5773502691896257;
137                 w[0]=1.0;
138                 w[1]=1.0;
139         }
140         if (m==3) {
141                 x[0]=−0.7745966692414834;
142                 x[1]=0.0;
143                 x[2]=0.7745966692414834;
144
145                 w[0]=0.5555555555555556;
146                 w[1]=0.8888888888888888;
147                 w[2]=0.5555555555555556;
148         }
149         if (m==4) {
150                 x[0]=−0.8611363115940526;
151                 x[1]=−0.3399810435848563;
152                 x[2]=0.3399810435848563;
153                 x[3]=0.8611363115940526;
154
155                 w[0]=0.3478548451374538;
156                 w[1]=0.6521451548625463;
157                 w[2]=0.6521451548625463;
158                 w[3]=0.3478548451374538;
159         }
160         if (m==5) {
161                 x[0]=−0.9061798459386641;
162                 x[1]=−0.5384693101056830;
163                 x[2]=0.0;
164                 x[3]=0.5384693101056830;
165                 x[4]=0.9061798459386641;
166
167                 w[0]=0.2369268850561892;
168                 w[1]=0.4786286704993664;
169                 w[2]=0.5688888888888889;
170                 w[3]=0.4786286704993664;
171                 w[4]=0.2369268850561892;
172         }
173 }
174
175 double cal_Aij1(int i, int k)
176 {
177         int k1,k2;
```

```
178         double sum=0.0;
179
180         k1=element[k][0]-1;
181         k2=element[k][1]-1;
182
183         double le=pow(pow(node[k1][0]-node[k2][0],2.0)+pow(node[k1][1]-node[k2
                ][1],2.0),0.5);
184
185             for (int a=0; a<NG ; a++)
186             {
187                     double Xuk=(1.0-x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[k2][0];
188                     double Yuk=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[k2][1];
189
190                     double fie=(1.0-x[a])/2.0;
191
192                     double F=(log(pow(node[i][0]-Xuk,2.0)+pow(node[i][1]-Yuk,2.0))-
                            log(pow(node[i][0]-Xuk,2.0)+pow(node[i][1]+Yuk,2.0)))/2.0;
193
194                     sum=sum+le*w[a]*fie*F;
195             }
196
197         return sum;
198 }
199
200 double cal_Aij2(int i, int k)
201 {
202         int k1,k2;
203         double sum=0.0;
204
205         k1=element[k][0]-1;
206         k2=element[k][1]-1;
207
208         double le=pow(pow(node[k1][0]-node[k2][0],2.0)+pow(node[k1][1]-node[k2
                ][1],2.0),0.5);
209
210         for (int a=0; a<NG ; a++)
211         {
212                 double Xuk=(1.0-x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[k2][0];
213                 double Yuk=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[k2][1];
214
215                 double fie=(1.0+x[a])/2.0;
216
217                 double F=(log(pow(node[i][0]-Xuk,2.0)+pow(node[i][1]-Yuk,2.0))-log(pow(
                        node[i][0]-Xuk,2.0)+pow(node[i][1]+Yuk,2.0)))/2.0;
218
```

```
219                     sum=sum+le*w[a]*fie*F;
220             }
221
222             return sum;
223     }
224
225     void DATAIN(const char name[], int nr, int nc, double ma[], double Aij[][N_node])
226     {
227             int i,j;
228             for (i=0; i<nr; i++)
229             {
230                     for (j=0; j<nc; j++)
231                     {
232                             ma[nc*i+j]=Aij[i][j];
233                     }
234             }
235     }
236
237     void DATAIN2(const char name[], int nr, int nc, double ma[], double V_node[])
238     {
239             int i,j;
240             for (i=0; i<nr; i++)
241             {
242                     for (j=0; j<nc; j++)
243                     {
244                             ma[nc*i+j]=V_node[i];
245                     }
246             }
247     }
248
249     int PIVOT(int *num, int nr, int k, double ma[] )
250     {
251             int i;
252             double aa,bb;
253
254             *num=k;
255             aa=fabs(ma[nr*k+k]);
256             for (i=k+1; i<nr; i++)
257             {
258                     if (fabs(ma[nr*i+k])>aa)
259                     {
260                             *num=i;
261                             aa=fabs(ma[nr*i+k]);
262                     }
263             }
```

```
264         if (fabs(aa)<=EPS) return 1;
265         if (*num==k) return 0;
266         for (i=k; i<nr; i++)
267         {
268                 bb=ma[nr*k+i];
269                 ma[nr*k+i]=ma[nr*(*num)+i];
270                 ma[nr*(*num)+i]=bb;
271         }
272         return 0;
273 }
274
275 int GAUSS(int nr, double ma[], double mb[], double mx[])
276 { int i,j,k;
277         int num;
278         double cc;
279
280         for (k=0; k<nr-1; k++)
281         {
282                 if (PIVOT(&num, nr, k, ma)!=0) return 1;
283                 if (num !=k)
284                 {
285                         cc=mb[num]; mb[num]=mb[k]; mb[k]=cc;
286                 }
287                 for (i=k+1; i<nr; i++)
288                 {
289                         cc=ma[nr*i+k]/ma[nr*k+k];
290                         for (j=k+1; j<nr; j++)
291                                 ma[nr*i+j]=ma[nr*i+j]-cc*ma[nr*k+j];
292                         mb[i]=mb[i]-cc*mb[k];
293                 }
294         }
295         for(k=nr-1;k>=0;k--)
296         {
297                 if (fabs(ma[nr*k+k])<=EPS) return 1;
298                 cc=0.0;
299                 for (j=k+1; j<nr; j++)
300                         cc+=ma[nr*k+j]*mx[j];
301                 mx[k]=(mb[k]-cc)/ma[nr*k+k];
302         }
303         return 0;
304 }
305
306 double get_boundary( double V_node[])
307 {
308         FILE *fp;
```

```
309         char lBuf[512],*p;
310
311         if ((fp = fopen("boundary_condition.txt","r")) == NULL){
312                 fprintf(stderr,"ファイルが開けません\n");
313                 return 1;
314         }
315         for(int m=0; m<N_node; m++ ) {
316                 fgets( lBuf, sizeof( lBuf ), fp );
317                 p = strtok( lBuf, "␣␣␣␣␣" );
318                 sscanf( p, "%lf", &V_node[m]);
319                 p = strtok( NULL, "␣␣␣␣␣" );
320         }
321         fclose(fp);
322         return 0;
323
324 }
325
326 void cal_imaginary(double charge_node[])
327 {
328         for (int k=0; k<N_ele; k++)
329         {
330                 for (int a=0; a<NG; a++)
331                 {
332                         int i,k1,k2;
333                         i=k*NG+a;
334                         k1=element[k][0]-1;
335                         k2=element[k][1]-1;
336
337                         double le=pow(pow(node[k1][0]-node[k2][0],2)+pow(node[k1][1]-
                                node[k2][1],2),0.5);
338
339                         charge_density[i]=le*w[a]*(charge_node[k1]*(1.0-x[a])/2.0+
                                charge_node[k2]*(1.0+x[a])/2.0);
340                 }
341         }
342 }
```

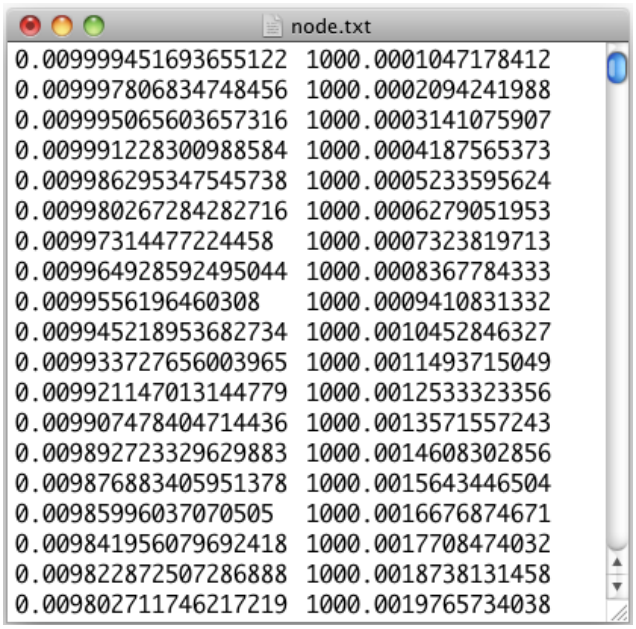


図 9.1: 2 次元表面電荷法の node.txt の例 (スクリーンショット)

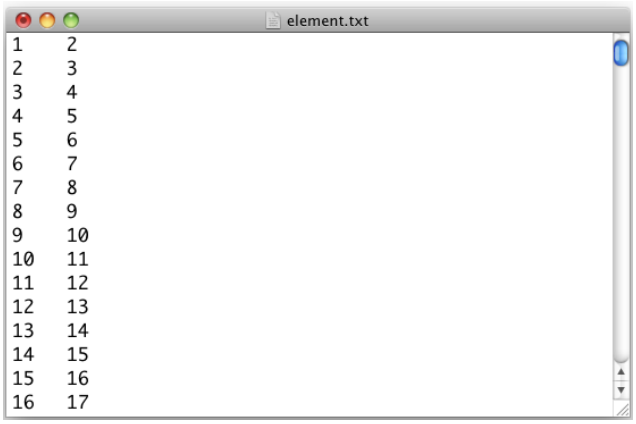


図 9.2: 2 次元表面電荷法の element.txt の例 (スクリーンショット)

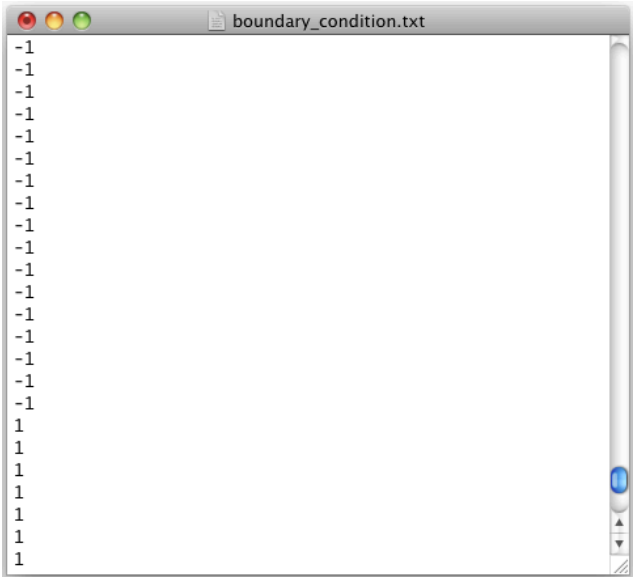


図 9.3: 2次元表面電荷法の boundary_condition.txt の例 (スクリーンショット)

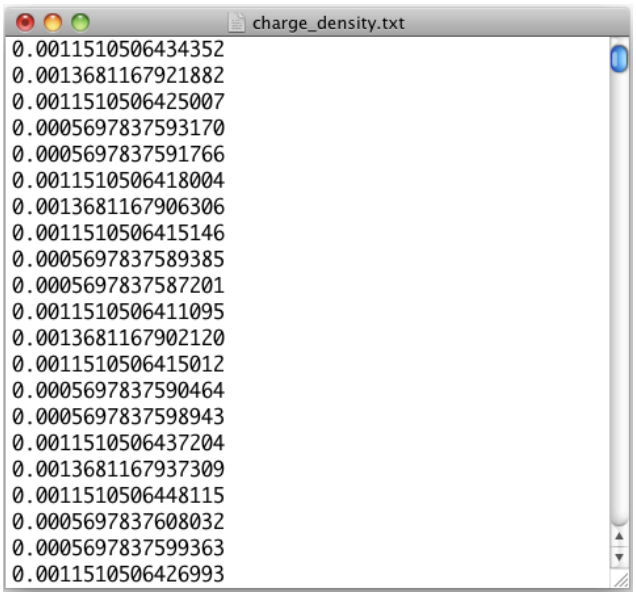


図 9.4: 2次元表面電荷法の charge_density.txt の例 (スクリーンショット)

ソースコード 9.2: field_2D.cpp（電場を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //各種の定数を定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 1500//要素数
9  #define N_node 1500//節点数
10 #define N_image NG*N_ele//仮想電荷の数
11 #define N_cal 25//計算点数
12
13 //グローバル変数
14 double charge_density[N_image];//仮想電荷（線電荷）の線電荷密度を格納
15 int element[N_ele][2];//要素の情報を格納
16 double node[N_node][2];//節点の情報を格納
17 double x[NG],w[NG];//積分点とその重みを格納
18
19 //関数プロトタイプ宣言
20 int get_element();//要素の情報を得る
21 int get_node();//節点の情報を得る
22 void gauss(int m);//ガウス積分公式の積分点、重みを決める
23 double get_calpoint(double keisanten[][2]);//計算点の情報を得る
24 double get_imaginary();//仮想電荷の電荷量を得る
25 void cal_field(double E[2],double keisanten[][2],int i, int n);//電場を計算する
26
27 //メイン関数
28 int main()
29 {
30         //各種情報を得る
31         get_element();
32         get_node();
33         gauss(NG);
34         get_imaginary();
35         double calpoint[N_cal][2];
36         get_calpoint(calpoint);
37
38         //電場の計算結果を書き込むファイルを準備
39         FILE *fp;
40         fp = fopen( "field.txt", "w" );
41         if( fp == NULL )
42         {
43                 puts( "ファイルが開けません" );
44                 return 1;
```

```
45          }
46
47          //電場を計算し、結果をファイルに書き込む
48          for (int i=0; i<N_cal ; i++)
49          {
50                  double E[2]={0.0,0.0};
51                  cal_field( E, calpoint, i, 0);
52                  fprintf( fp, "%16.16lf␣", E[0] );
53                  fprintf( fp, "%16.16lf␣", E[1] );
54                  fprintf( fp, "\n" );
55          }
56          fclose( fp );
57          return 0;
58  }
59
60  //関数の定義
61  int get_node()
62  {
63          FILE *fp;
64          char lBuf[512],*p;
65
66          if ((fp = fopen("node.txt","r")) == NULL){
67                  fprintf(stderr,"ファイルが開けません\n");
68                  return 1;
69          }
70          for(int m=0; m<N_node; m++ ) {
71                  fgets( lBuf, sizeof( lBuf ), fp );
72                  p = strtok( lBuf, "␣␣␣␣␣" );
73                  for (int n=0; n<2 && p!=NULL; n++) {
74                          sscanf( p, "%lf", &node[m][n] );
75                          p = strtok( NULL, "␣␣␣␣␣" );
76                  }
77          }
78          fclose(fp);
79          return 0;
80  }
81
82  int get_element()
83  {
84          FILE *fp;
85          char lBuf[512],*p;
86
87          if ((fp = fopen("element.txt","r")) == NULL){
88                  fprintf(stderr,"ファイルが開けません\n");
89                  return 1;
```

```
90          }
91          for(int m=0; m<N_ele; m++ ) {
92                  fgets( lBuf, sizeof( lBuf ), fp );
93                  p = strtok( lBuf, "␣␣␣␣␣" );
94                  for (int n=0; n<2 && p!=NULL; n++) {
95                          sscanf( p, "%d", &element[m][n] );
96                          p = strtok( NULL, "␣␣␣␣␣" );
97                  }
98          }
99          fclose(fp);
100         return 0;
101 }
102
103 void gauss(int m)
104 {
105         if (m==2) {
106                 x[0]=-0.5773502691896257;
107                 x[1]=0.5773502691896257;
108                 w[0]=1.0;
109                 w[1]=1.0;
110         }
111         if (m==3) {
112                 x[0]=-0.7745966692414834;
113                 x[1]=0.0;
114                 x[2]=0.7745966692414834;
115
116                 w[0]=0.5555555555555556;
117                 w[1]=0.8888888888888888;
118                 w[2]=0.5555555555555556;
119         }
120         if (m==4) {
121                 x[0]=-0.8611363115940526;
122                 x[1]=-0.3399810435848563;
123                 x[2]=0.3399810435848563;
124                 x[3]=0.8611363115940526;
125
126                 w[0]=0.3478548451374538;
127                 w[1]=0.6521451548625463;
128                 w[2]=0.6521451548625463;
129                 w[3]=0.3478548451374538;
130         }
131         if (m==5) {
132                 x[0]=-0.9061798459386641;
133                 x[1]=-0.5384693101056830;
134                 x[2]=0.0;
```

```
135                     x[3]=0.5384693101056830;
136                     x[4]=0.9061798459386641;
137
138                     w[0]=0.2369268850561892;
139                     w[1]=0.4786286704993664;
140                     w[2]=0.5688888888888889;
141                     w[3]=0.4786286704993664;
142                     w[4]=0.2369268850561892;
143             }
144     }
145
146     double get_imaginary()
147     {
148             FILE *fp2;
149             char lBuf_2[512],*p2;
150             if ((fp2 = fopen("charge_density.txt","r")) == NULL){
151                     fprintf(stderr,"ファイルが開けません\n");
152                     return 1;
153             }
154             for(int m=0; m<N_image ; m++ ) {
155                     fgets( lBuf_2, sizeof( lBuf_2 ), fp2 );
156                     p2 = strtok( lBuf_2, "␣␣" );
157                     sscanf( p2, "%lf", &charge_density[m]);
158                     p2 = strtok( NULL, "␣␣␣␣" );
159             }
160             fclose(fp2);
161             return 0;
162     }
163
164     void cal_field(double E[2],double calpoint[][2],int i, int n)
165     {
166             for (int k=0; k<N_ele; k++)
167             {
168                     for (int a=0; a<NG; a++)
169                     {
170                             int t,k1,k2;
171                             t=k*NG+a;
172                             k1=element[k][0]-1;
173                             k2=element[k][1]-1;
174
175                             double Xuk=(1.0-x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[k2][0];
176                             double Yuk=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[k2][1];
177
178                             double Lp=pow(calpoint[i][0]-Xuk,2.0)+pow(calpoint[i][1]-Yuk,2.0);
179                             double Lm=pow(calpoint[i][0]-Xuk,2.0)+pow(calpoint[i][1]+Yuk
```

```
                                ,2.0);
180
181                             E[0] += (charge_density[t])*(calpoint[i][0]-Xuk)*(1.0/Lm-1.0/Lp);
182
183                             E[1] += (charge_density[t])*((calpoint[i][1]+Yuk)/Lm-(calpoint[i][1]-
                                    Yuk)/Lp);
184                     }
185             }
186     }
187
188     double get_calpoint(double calpoint[][2])
189     {
190             FILE *fp;
191             char lBuf[512],*p;
192
193             if ((fp = fopen("calculation_point.txt","r")) == NULL){
194                     fprintf(stderr,"ファイルが開けません\n");
195                     return 1;
196             }
197
198             for(int m=0; m<N_cal; m++ ) {
199                     fgets( lBuf, sizeof( lBuf ), fp );
200
201                     p = strtok( lBuf, "␣␣␣␣␣" );
202                     for(int n=0;n<2;n++){
203                             sscanf( p, "%lf", &calpoint[m][n]);
204                             p = strtok( NULL, "␣␣␣␣␣" );
205                     }
206             }
207             fclose(fp);
208             return 0;
209     }
```

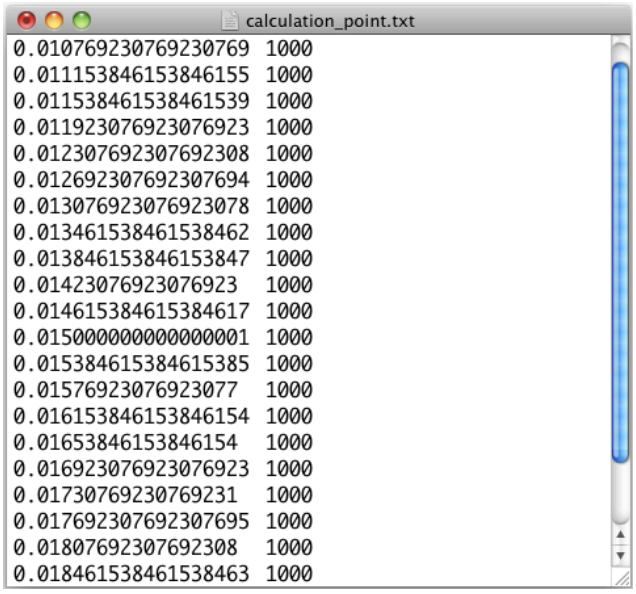


図 9.5: 2 次元表面電荷法の calculation_point.txt の例 (スクリーンショット)

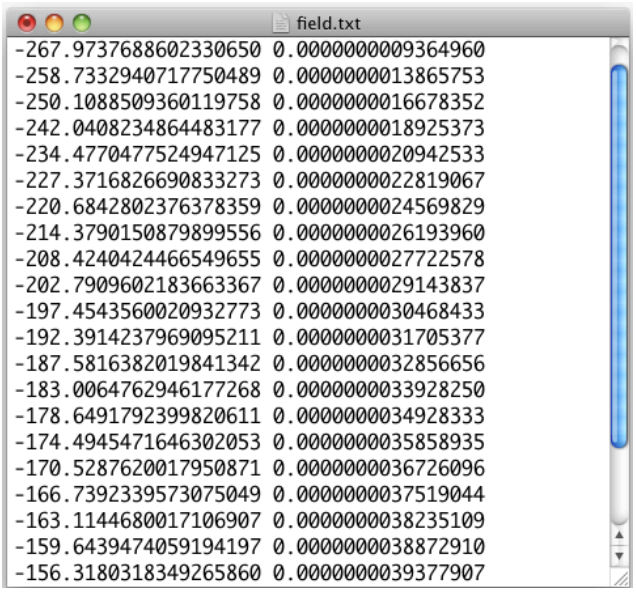


図 9.6: 2 次元表面電荷法の field.txt の例 (スクリーンショット)

9.2　回転対称場の表面電荷法プログラム

仮想電荷（リング電荷）の線電荷密度を計算するC++プログラムimaginary_rotation.cpp をソースコード9.3に示す。また、電場を計算するC++プログラムfield_rotation.cpp をソースコード9.4に示す。

imaginary_rotation.cppは電極の形状と境界条件を指定すれば、仮想電荷の電荷量を計算する。２次元場のときと同様で、電極形状の指定は入力ファイルnode.txtとelement.txtで行い、境界条件の指定は入力ファイルboundary_condition.txtで行う。回転対称場の場合のnode.txtは１列目, ２列目の数値はそれぞれ節点のr,z座標を入力する。element.txtとboundary_condition.txtはは２次元場のときと同様である。このような入力ファイルを作成し、imaginary_rotation.cppを実行すると、出力ファイルcharge_density.txtが出力される。charge_density.txtは図9.4と同様な１列の数値が並んだデータ形式であり、仮想電荷（リング電荷）の線電荷密度を表す。

そして、field_rotation.cppを実行すれば、charge_density.txtとcaluculation_point.txtを読み込んで回転対称な電場を計算する。計算結果はとfield.txtして出力される。caluculation_ point.txtは２列の数値データで、１列目, ２列目の数値はそれぞれ電場を計算する点のr,z座標を入力する。出力ファイルfield.txtは２列の数値データで、１列目, ２列目の数値はそれぞれr方向の電場,z方向の電場を表す。

ソースコード 9.3: imaginary_rotation.cpp（仮想電荷の電荷量を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //各種の定数を定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 1998//要素数
9  #define N_node 2000//節点数
10 #define N_image NG*N_ele//仮想線電荷の数
11 #define EPS 1.0e-5//ガウス消去法の許容誤差
12
13 //グローバル変数
14 int element[N_ele][2];//要素の情報を格納
15 double node[N_node][2];//節点の情報を格納
16 double x[NG],w[NG];//積分点とその重みを格納
17 double charge_density[N_image];//仮想電荷（リング電荷）の線電荷密度を格納
18
19 //関数のプロトタイプ宣言
20 int get_element();//要素の情報を得る
21 int get_node();//節点の情報を得る
22 double get_boundary( double V_node[]);//境界条件の情報を得る
23 void gauss(int m);//ガウス積分公式の積分点、重みを決める
24 double cal_Aij1(int i, int k);//電位係数の計算 1
25 double cal_Aij2(int i, int k);//電位係数の計算 2
26 void cal_imaginary(double charge_node[]);//仮想電荷の電荷量を計算する
27
28 //ガウス消去法に用いる関数のプロトタイプ宣言
29 void DATAIN(const char name[], int nr, int nc,double ma[], double Aij[][N_node]);//電位
       係数の行列を作成
30 void DATAIN2(const char name[], int nr, int nc, double ma[], double V_node[]);//境界条件
       の行列を作成
31 int PIVOT(int *num, int nr, int k, double ma[]);//ピボットの選択
32 int GAUSS(int nr, double ma[], double mb[], double mx[]);//ガウス消去法
33
34 //メイン関数
35 int main()
36 {
37         //要素、節点の情報を得る
38         get_element();
39         get_node();
40
41         //ガウス積分の積分点と重みを決める
```

```
42          gauss(NG);
43
44          //節点の電位係数の計算
45          static double Aij[N_node][N_node]={0.0};
46          for (int k=0; k<N_ele ; k++)
47          {
48                  for (int i=0; i<N_node; i++)
49                  {
50                          int k1,k2;
51                          k1=element[k][0]-1; k2=element[k][1]-1;
52
53                          Aij[i][k1] += cal_Aij1(i,k);
54                          Aij[i][k2] += cal_Aij2(i,k);
55                  }
56          }
57
58          //節点の線電荷密度を計算する
59          int nr=N_node,nc;
60          int flag;
61          static double ma[N_node*N_node];
62          double mb[N_node],charge_node[N_node];
63          nc=nr;
64          DATAIN("deni_keisu",nr,nc,ma,Aij);
65          double V_node[N_node];
66          get_boundary(V_node);
67          DATAIN2("V_electrode",nr,1,mb,V_node);
68          flag=GAUSS(nr,ma,mb,charge_node);
69          if(flag) printf("\n 計算不能\n");
70
71          //仮想電荷（リング電荷）の線電荷密度を計算する
72          cal_imaginary(charge_node);
73
74          //仮想電荷の電荷量をテキストデータに出力
75          FILE *fp;
76          fp = fopen( "charge_density.txt", "w" );
77          if( fp == NULL )
78          {
79                  puts( "ファイルが開けません" );
80                  return 1;
81          }
82          for(int j=0; j<N_image; ++j)
83          {
84                  fprintf( fp, "%16.16lf", charge_density[j]);
85                  fprintf( fp, "\n" );
86          }
```

```
87          fclose( fp );
88
89          return 0;
90  }
91
92  //関数の定義
93  int get_node()
94  {
95          FILE *fp;
96          char lBuf[512],*p;
97
98          if ((fp = fopen("node.txt","r")) == NULL){
99                  fprintf(stderr,"ファイルが開けません\n");
100                 return 1;
101         }
102         for(int m=0; m<N_node; m++ ) {
103                 fgets( lBuf, sizeof( lBuf ), fp );
104                 p = strtok( lBuf, "␣␣␣␣␣" );
105                 for (int n=0; n<2 && p!=NULL; n++) {
106                         sscanf( p, "%lf", &node[m][n] );
107                         p = strtok( NULL, "␣␣␣␣␣" );
108                 }
109         }
110         fclose(fp);
111         return 0;
112 }
113
114 int get_element()
115 {
116         FILE *fp;
117         char lBuf[512],*p;
118
119         if ((fp = fopen("element.txt","r")) == NULL){
120                 fprintf(stderr,"ファイルが開けません\n");
121                 return 1;
122         }
123         for(int m=0; m<N_ele; m++ ) {
124                 fgets( lBuf, sizeof( lBuf ), fp );
125                 p = strtok( lBuf, "␣␣␣␣␣" );
126                 for (int n=0; n<2 && p!=NULL; n++) {
127                         sscanf( p, "%d", &element[m][n] );
128                         p = strtok( NULL, "␣␣␣␣␣" );
129                 }
130         }
131         fclose(fp);
```

```
132         return 0;
133 }
134
135 void gauss(int m)
136 {
137         if (m==2) {
138                 x[0]=-0.5773502691896257;
139                 x[1]=0.5773502691896257;
140                 w[0]=1.0;
141                 w[1]=1.0;
142         }
143         if (m==3) {
144                 x[0]=-0.7745966692414834;
145                 x[1]=0.0;
146                 x[2]=0.7745966692414834;
147
148                 w[0]=0.5555555555555556;
149                 w[1]=0.8888888888888888;
150                 w[2]=0.5555555555555556;
151         }
152         if (m==4) {
153                 x[0]=-0.8611363115940526;
154                 x[1]=-0.3399810435848563;
155                 x[2]=0.3399810435848563;
156                 x[3]=0.8611363115940526;
157
158                 w[0]=0.3478548451374538;
159                 w[1]=0.6521451548625463;
160                 w[2]=0.6521451548625463;
161                 w[3]=0.3478548451374538;
162         }
163         if (m==5) {
164                 x[0]=-0.9061798459386641;
165                 x[1]=-0.5384693101056830;
166                 x[2]=0.0;
167                 x[3]=0.5384693101056830;
168                 x[4]=0.9061798459386641;
169
170                 w[0]=0.2369268850561892;
171                 w[1]=0.4786286704993664;
172                 w[2]=0.5688888888888889;
173                 w[3]=0.4786286704993664;
174                 w[4]=0.2369268850561892;
175         }
176 }
```

```
177
178 double cal_Aij1(int i, int k)
179 {
180         int k1,k2;
181         double sum=0.0;
182
183         k1=element[k][0]-1;
184         k2=element[k][1]-1;
185
186         double le=sqrt(pow(node[k1][0]-node[k2][0],2.0)+pow(node[k1][1]-node[k2][1],2.0));
187
188                 for (int a=0; a<NG ; a++)
189                 {
190                         double Ru=(1.0-x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[k2][0];
191                         double Zu=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[k2][1];
192
193                         double r=node[i][0];
194                         double z=node[i][1];
195
196                         double kk1=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z-Zu,2.0)));
197                         double kk2=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z+Zu,2.0)));
198                         double a1[5],b1[5],a2[5],b2[5];
199                         a1[0]=1.0;
200                         b1[0]=sqrt(1-pow(kk1,2.0));
201                         a2[0]=1.0;
202                         b2[0]=sqrt(1-pow(kk2,2.0));
203                         for (int n=0; n<4; n++) {
204                                 a1[n+1]=(a1[n]+b1[n])/2.0;
205                                 b1[n+1]=sqrt(a1[n]*b1[n]);
206                                 a2[n+1]=(a2[n]+b2[n])/2.0;
207                                 b2[n+1]=sqrt(a2[n]*b2[n]);
208                         }
209                         double ellip11=M_PI/(2.0*a1[4]);
210                         double ellip12=M_PI/(2.0*a2[4]);
211
212                         double F=ellip11/sqrt(pow(r+Ru,2.0)+pow(z-Zu,2.0))-ellip12/sqrt(
                                 pow(r+Ru,2.0)+pow(z+Zu,2.0));
213                         double fie=(1.0-x[a])/2.0;
214                         sum+=le*w[a]*fie*F;
215
216                 }
217         return sum;
218 }
219
220 double cal_Aij2(int i, int k)
```

```
221 {
222         int k1,k2;
223         double sum=0.0;
224 
225         k1=element[k][0]-1;
226         k2=element[k][1]-1;
227 
228         double le=pow(pow(node[k1][0]-node[k2][0],2.0)+pow(node[k1][1]-node[k2
                ][1],2.0),0.5);
229 
230         for (int a=0; a<NG ; a++)
231         {
232                 double Ru=(1.0-x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[k2][0];
233                 double Zu=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[k2][1];
234 
235                 double r=node[i][0];
236                 double z=node[i][1];
237 
238                 double kk1=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z-Zu,2.0)));
239                 double kk2=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z+Zu,2.0)));
240                 double a1[5],b1[5],a2[5],b2[5];
241                 a1[0]=1.0;
242                 b1[0]=sqrt(1-pow(kk1,2.0));
243                 a2[0]=1.0;
244                 b2[0]=sqrt(1-pow(kk2,2.0));
245                 for (int n=0; n<4; n++) {
246                         a1[n+1]=(a1[n]+b1[n])/2.0;
247                         b1[n+1]=sqrt(a1[n]*b1[n]);
248                         a2[n+1]=(a2[n]+b2[n])/2.0;
249                         b2[n+1]=sqrt(a2[n]*b2[n]);
250                 }
251                 double ellip11=M_PI/(2.0*a1[4]);
252                 double ellip12=M_PI/(2.0*a2[4]);
253 
254                 double F=ellip11/sqrt(pow(r+Ru,2.0)+pow(z-Zu,2.0))-ellip12/sqrt(pow(r+
                        Ru,2.0)+pow(z+Zu,2.0));
255                 double fie=(1.0+x[a])/2.0;
256                 sum+=le*w[a]*fie*F;
257 
258         }
259         return sum;
260 }
261 
262 void DATAIN(const char name[], int nr, int nc, double ma[], double Aij[][N_node])
263 {
```

```
264         int i,j;
265         for (i=0; i<nr; i++)
266         {
267                 for (j=0; j<nc; j++)
268                 {
269                         ma[nc*i+j]=Aij[i][j];
270                 }
271         }
272 }
273
274 void DATAIN2(const char name[], int nr, int nc, double ma[], double V_node[])
275 {
276         int i,j;
277         for (i=0; i<nr; i++)
278         {
279                 for (j=0; j<nc; j++)
280                 {
281                         ma[nc*i+j]=V_node[i];
282                 }
283         }
284 }
285
286 int PIVOT(int *num, int nr, int k, double ma[] )
287 {
288         int i;
289         double aa,bb;
290
291         *num=k;
292         aa=fabs(ma[nr*k+k]);
293         for (i=k+1; i<nr; i++)
294         {
295                 if (fabs(ma[nr*i+k])>aa)
296                 {
297                         *num=i;
298                         aa=fabs(ma[nr*i+k]);
299                 }
300         }
301         if (fabs(aa)<=EPS) return 1;
302         if (*num==k) return 0;
303         for (i=k; i<nr; i++)
304         {
305                 bb=ma[nr*k+i];
306                 ma[nr*k+i]=ma[nr*(*num)+i];
307                 ma[nr*(*num)+i]=bb;
308         }
```

```
309         return 0;
310 }
311
312 int GAUSS(int nr, double ma[], double mb[], double mx[])
313 { int i,j,k;
314         int num;
315         double cc;
316
317         for (k=0; k<nr-1; k++)
318         {
319                 if (PIVOT(&num, nr, k, ma)!=0) return 1;
320                 if (num !=k)
321                 {
322                         cc=mb[num]; mb[num]=mb[k]; mb[k]=cc;
323                 }
324                 for (i=k+1; i<nr; i++)
325                 {
326                         cc=ma[nr*i+k]/ma[nr*k+k];
327                         for (j=k+1; j<nr; j++)
328                                 ma[nr*i+j]=ma[nr*i+j]-cc*ma[nr*k+j];
329                         mb[i]=mb[i]-cc*mb[k];
330                 }
331         }
332         for(k=nr-1;k>=0;k--)
333         {
334                 if (fabs(ma[nr*k+k])<=EPS) return 1;
335                 cc=0.0;
336                 for (j=k+1; j<nr; j++)
337                         cc+=ma[nr*k+j]*mx[j];
338                 mx[k]=(mb[k]-cc)/ma[nr*k+k];
339         }
340         return 0;
341 }
342
343 double get_boundary( double V_node[])
344 {
345         FILE *fp;
346         char lBuf[512],*p;
347
348         if ((fp = fopen("boundary_condition.txt","r")) == NULL){
349                 fprintf(stderr,"ファイルが開けません\n");
350                 return 1;
351         }
352         for(int m=0; m<N_node; m++ ) {
353                 fgets( lBuf, sizeof( lBuf ), fp );
```

```
354                     p = strtok( lBuf, "␣␣␣␣␣" );
355                     sscanf( p, "%lf", &V_node[m]);
356                     p = strtok( NULL, "␣␣␣␣␣" );
357             }
358             fclose(fp);
359             return 0;
360     }
361
362     void cal_imaginary(double charge_node[])
363     {
364             for (int k=0; k<N_ele; k++)
365             {
366                     for (int a=0; a<NG; a++)
367                     {
368                             int i,k1,k2;
369                             i=k*NG+a;
370                             k1=element[k][0]-1;
371                             k2=element[k][1]-1;
372
373                             double le=sqrt(pow(node[k1][0]-node[k2][0],2.0)+pow(node[k1][1]-
                                    node[k2][1],2.0));
374
375                             charge_density[i]=le*w[a]*(charge_node[k1]*(1.0-x[a])/2.0+
                                    charge_node[k2]*(1.0+x[a])/2.0);
376                     }
377             }
378     }
```

ソースコード 9.4: field_rotation.cpp（電場を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //各種の定数を定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 1998//要素数
9  #define N_node 2000//節点数
10 #define N_image NG*N_ele//仮想線電荷の数
11 #define N_cal 100//計算点数
12
13 //グローバル変数
14 double charge_density[N_image];//仮想電荷（リング電荷）の線電荷密度
15 int element[N_ele][2];//要素の情報を格納
16 double node[N_node][2];//節点の情報を格納
17 double x[NG],w[NG];//積分点とその重みを格納
18
19 //関数プロトタイプ宣言
20 int get_element();//要素の情報を得る
21 int get_node();//節点の情報を得る
22 void gauss(int m);//ガウス積分公式の積分点、重みを決める
23 double get_calpoint(double calpoint[][2]);//計算点の情報を得る
24 double get_imaginary();//仮想点電荷の電荷量を得る
25 double get_calpoint(double calpoint[][2]);//計算点の情報を得る
26 void cal_field(double E[][2],double calpoint[][2], int i);//計算点の電場を計算する
27
28 //メイン
29 int main()
30 {
31         //各種情報を得る
32         get_element();
33         get_node();
34         gauss(NG);
35         get_imaginary();
36         double calpoint[N_cal][2];
37         get_calpoint(calpoint);
38
39         //電場の計算結果を書き込むファイルを準備
40         FILE *fp;
41         fp = fopen( "field.txt", "w" );
42         if( fp == NULL )
43         {
44                 puts( "ファイルが開けません" );
```

```
45                  return 1;
46          }
47
48          //電場を計算し、結果をファイルに書き込む
49          double E[N_cal][2]={0.0};
50          for(int i=0; i<N_cal; ++i)
51          {
52                  cal_field( E, calpoint, i);
53                  fprintf( fp, "%16.16lf␣", E[i][0]);
54                  fprintf( fp, "%16.16lf␣", E[i][1]);
55                  fprintf( fp, "\n" );
56          }
57          fclose( fp );
58
59          return 0;
60  }
61
62  //関数の定義
63  int get_node()
64  {
65          FILE *fp;
66          char lBuf[512],*p;
67
68          if ((fp = fopen("node.txt","r")) == NULL){
69                  fprintf(stderr,"ファイルが開けません\n");
70                  return 1;
71          }
72          for(int m=0; m<N_node; m++ ) {
73                  fgets( lBuf, sizeof( lBuf ), fp );
74                  p = strtok( lBuf, "␣␣␣␣␣" );
75                  for (int n=0; n<2 && p!=NULL; n++) {
76                          sscanf( p, "%lf", &node[m][n] );
77                          p = strtok( NULL, "␣␣␣␣␣" );
78                  }
79          }
80          fclose(fp);
81          return 0;
82  }
83
84  int get_element()
85  {
86          FILE *fp;
87          char lBuf[512],*p;
88
89          if ((fp = fopen("element.txt","r")) == NULL){
```

```
90                  fprintf(stderr,"ファイルが開けません\n");
91                  return 1;
92          }
93          for(int m=0; m<N_ele; m++ ) {
94                  fgets( lBuf, sizeof( lBuf ), fp );
95                  p = strtok( lBuf, "␣␣␣␣␣" );
96                  for (int n=0; n<2 && p!=NULL; n++) {
97                          sscanf( p, "%d", &element[m][n] );
98                          p = strtok( NULL, "␣␣␣␣␣" );
99                  }
100         }
101         fclose(fp);
102         return 0;
103 }
104
105 void gauss(int m)
106 {
107         if (m==2) {
108                 x[0]=−0.5773502691896257;
109                 x[1]=0.5773502691896257;
110                 w[0]=1.0;
111                 w[1]=1.0;
112         }
113         if (m==3) {
114                 x[0]=−0.7745966692414834;
115                 x[1]=0.0;
116                 x[2]=0.7745966692414834;
117
118                 w[0]=0.5555555555555556;
119                 w[1]=0.8888888888888888;
120                 w[2]=0.5555555555555556;
121         }
122         if (m==4) {
123                 x[0]=−0.8611363115940526;
124                 x[1]=−0.3399810435848563;
125                 x[2]=0.3399810435848563;
126                 x[3]=0.8611363115940526;
127
128                 w[0]=0.3478548451374538;
129                 w[1]=0.6521451548625463;
130                 w[2]=0.6521451548625463;
131                 w[3]=0.3478548451374538;
132         }
133         if (m==5) {
134                 x[0]=−0.9061798459386641;
```

```
135                     x[1]=−0.5384693101056830;
136                     x[2]=0.0;
137                     x[3]=0.5384693101056830;
138                     x[4]=0.9061798459386641;
139
140                     w[0]=0.2369268850561892;
141                     w[1]=0.4786286704993664;
142                     w[2]=0.5688888888888889;
143                     w[3]=0.4786286704993664;
144                     w[4]=0.2369268850561892;
145             }
146     }
147
148     double get_imaginary()
149     {
150             FILE *fp2;
151             char lBuf_2[512],*p2;
152
153             if ((fp2 = fopen("charge_density.txt","r")) == NULL){
154                     fprintf(stderr,"ファイルが開けません\n");
155                     return 1;
156             }
157             for(int m=0; m<N_image ; m++ ) {
158                     fgets( lBuf_2, sizeof( lBuf_2 ), fp2 );
159                     p2 = strtok( lBuf_2, "␣␣" );
160                     sscanf( p2, "%lf", &charge_density[m]);
161                     p2 = strtok( NULL, "␣␣␣␣" );
162
163             }
164             fclose(fp2);
165             return 0;
166     }
167
168     void cal_field(double E[][2],double calpoint[][2], int i)
169     {
170             for (int k=0; k<N_ele; k++)
171                     {
172                             for (int a=0; a<NG; a++)
173                             {
174                                     int t,k1,k2;
175                                     t=k*NG+a;
176                                     k1=element[k][0]−1;
177                                     k2=element[k][1]−1;
178
179                                     double Ru=(1.0−x[a])/2.0*node[k1][0]+(1.0+x[a])/2.0*node[
```

```
        k2][0];
180     double Zu=(1.0-x[a])/2.0*node[k1][1]+(1.0+x[a])/2.0*node[
        k2][1];
181
182     double r=calpoint[i][0];
183     double z=calpoint[i][1];
184
185     double kk1=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z-Zu
        ,2.0)));
186     double kk2=sqrt(4.0*r*Ru/(pow(r+Ru,2.0)+pow(z+Zu
        ,2.0)));
187
188     double a1[5],b1[5];
189     double a2[5],b2[5];
190     a1[0]=1.0;
191     b1[0]=sqrt(1-pow(kk1,2.0));
192     a2[0]=1.0;
193     b2[0]=sqrt(1-pow(kk2,2.0));
194     double sum1=0.0,sum2=0.0;
195     for (int n=0; n<4; n++) {
196         a1[n+1]=(a1[n]+b1[n])/2.0;
197         b1[n+1]=sqrt(a1[n]*b1[n]);
198         a2[n+1]=(a2[n]+b2[n])/2.0;
199         b2[n+1]=sqrt(a2[n]*b2[n]);
200     }
201     for (int n=0; n<5; n++) {
202         sum1+=pow(2.0,n-1)*(pow(a1[n],2.0)-pow(b1[n
            ],2.0));
203         sum2+=pow(2.0,n-1)*(pow(a2[n],2.0)-pow(b2[n
            ],2.0));
204     }
205     double ellipK1=M_PI/(2.0*a1[4]);
206     double ellipK2=M_PI/(2.0*a2[4]);
207     double ellipE1=M_PI*(1.0-sum1)/(2.0*a1[4]);
208     double ellipE2=M_PI*(1.0-sum2)/(2.0*a2[4]);
209
210     double FEr=-(((pow(Ru,2.0)-pow(r,2.0)+pow(z-Zu,2.0))*
        ellipE1-(pow(r-Ru,2.0)+pow(z-Zu,2.0))*ellipK1)/(2.0*
        r*sqrt(pow(r+Ru,2.0)+pow(z-Zu,2.0))*(pow(r-Ru
        ,2.0)+pow(z-Zu,2.0)))
211                                 -((pow(Ru,2.0)-pow(r,2.0)+pow(z
                                        +Zu,2.0))*ellipE2-(pow(r-Ru
                                        ,2.0)+pow(z+Zu,2.0))*ellipK2
                                        )/(2.0*r*sqrt(pow(r+Ru,2.0)+
                                        pow(z+Zu,2.0))*(pow(r-Ru
```

```
                                                                    ,2.0)+pow(z+Zu,2.0))));
212
213
214                                 double FEz=(z-Zu)*ellipE1/(sqrt(pow(r+Ru,2.0)+pow(z-
                                        Zu,2.0))*(pow(r-Ru,2.0)+pow(z-Zu,2.0)))
215                                                                 -((z+Zu)*ellipE2/(sqrt(pow(
                                                                        r+Ru,2.0)+pow(z+Zu
                                                                        ,2.0))*(pow(r-Ru,2.0)+
                                                                        pow(z+Zu,2.0))));
216
217                                 E[i][0]+=charge_density[t]*FEr;
218                                 E[i][1]+=charge_density[t]*FEz;
219                             }
220                     }
221 }
222
223 double get_calpoint(double calpoint[][2])
224 {
225         FILE *fp;
226         char lBuf[512],*p;
227
228         if ((fp = fopen("calculation_point.txt","r")) == NULL){
229                 fprintf(stderr,"ファイルが開けません\n");
230                 return 1;
231         }
232         for(int m=0; m<N_cal; m++ ) {
233                 fgets( lBuf, sizeof( lBuf ), fp );
234
235                 p = strtok( lBuf, "␣␣␣␣␣" );
236
237                 for(int n=0;n<2;n++){
238                         sscanf( p, "%lf", &calpoint[m][n]);
239                         p = strtok( NULL, "␣␣␣␣␣" );
240                 }
241         }
242         fclose(fp);
243
244         return 0;
245 }
```

9.3　３次元場の表面電荷法プログラム

仮想電荷（点電荷）の電荷量を計算する C++プログラム imaginary_3D.cpp をソースコード 9.5 に示す。また、電場を計算する C++プログラム field_3D.cpp をソースコード 9.6 に示す。

imaginary_3D.cpp は電極の形状と境界条件を指定すれば、仮想電荷の電荷量を計算する。電極形状の指定は入力ファイル node.txt と element.txt で行い、境界条件の指定は入力ファイル boundary_condition.txt で行う。３次元場の場合の node.txt は 3 列の数値データであり、１列目, ２列目,3 列目の数値はそれぞれ節点の x,y,z 座標を入力する。element.txt は 3 列の数値データであり、1 列目,2 列目,3 列目に正数を入力し、要素を構成する３つの節点の番号を各行で指定する。boundary_condition.txt は２次元場, 回転対称場のときと同様である。このような入力ファイルを作成し、imaginary_3D.cpp を実行すると、出力ファイル charge_quantity.txt が出力される。charge_quantity.txt は１列の数値が並んだデータ形式であり、仮想電荷（点電荷）の電荷量を表す。

そして、field_3D.cpp を実行すれば、charge_quantity.txt と caluculation_point.txt を読み込んで３次元の電場を計算する。計算結果は field.txt として出力される。caluculation_ point.txt は 3 列の数値データで、1 列目,2 列目,3 列目の数値はそれぞれ電場を計算する点の x,y,z 座標を表す。field.txt は 3 列の数値データで、１列目, ２列目,3 列目の数値はそれぞれ x,y,z 方向の電場を表す。

ソースコード 9.5: imaginary_3D.cpp（仮想電荷の電荷量を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //各種の定数を定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 3364//要素数
9  #define N_node 1740//節点数
10 #define N_image N_ele*25//ガウス消去法の許容誤差
11 #define EPS 1.0e-5//ガウス消去法の誤差の限界
12
13 //グローバル変数
14 int element[N_ele][3];//要素の情報を格納
15 double node[N_node][3];//節点の情報を格納
16 double space[N_ele];//要素の面積を格納
17 double x[NG],w[NG];//積分点とその重みを格納
18 double charge_quantity[N_image];//仮想電荷（点電荷）の電荷量を格納
19
20 //関数のプロトタイプ宣言
21 int get_element();//要素の情報を得る
22 int get_node();//節点の情報を得る
23 double get_boundary(double V_node[]);//境界条件の情報を得る
24 void cal_space();//要素の面積を計算
25 void gauss(int m);//ガウス積分公式の積分点、重みを決める
26 double cal_Aij1(int i, int k);//電位係数の計算 1
27 double cal_Aij2(int i, int k);//電位係数の計算 2
28 double cal_Aij3(int i, int k);//電位係数の計算 3
29 void cal_imaginary(double charge_node[]);//仮想電荷の電荷量を計算する
30
31 //ガウス消去法に用いる関数のプロトタイプ宣言
32 void DATAIN(const char name[], int nr, int nc,double ma[],double Aij[][N_node]);//電位係
       数の行列を作成
33 void DATAIN2(const char name[], int nr, int nc,double ma[],double V_node[]);//境界条件
       の行列を作成
34 int PIVOT(int *num, int nr, int k,double ma[]);//ピボットの選択
35 int GAUSS(int nr,double ma[],double mb[],double mx[]);//ガウス消去法
36
37 //メイン関数
38 int main()
39 {
40         //要素、節点の情報を得る
41         get_element();
42         get_node();
```

```
43          cal_space();
44
45          //ガウス積分の積分点と重みを決める
46          gauss(NG);
47
48          //節点の電位係数の計算
49          static double Aij[N_node][N_node]={0};
50          for (int k=0; k<N_ele ; k++)
51          {
52                  for (int i=0; i<N_node; i++)
53                  {
54                          int k1,k2,k3;
55                          k1=element[k][0]-1; k2=element[k][1]-1; k3=element[k][2]-1;
56
57                          Aij[i][k1] += cal_Aij1(i,k);
58                          Aij[i][k2] += cal_Aij2(i,k);
59                          Aij[i][k3] += cal_Aij3(i,k);
60                  }
61          }
62
63          //節点の線電荷密度を計算する
64          int nr=N_node,nc;
65          int flag;
66          static double ma[N_node*N_node];
67          double mb[N_node],charge_node[N_node];
68          nc=nr;
69          DATAIN("deni_keisu",nr,nc,ma,Aij);
70          double V_node[N_node];
71          get_boundary(V_node);
72          DATAIN2("V_electrode",nr,1,mb,V_node);
73          flag=GAUSS(nr,ma,mb,charge_node);
74          if(flag) printf("\n 計算不能\n");
75
76          //仮想電荷（点電荷）の電荷量を計算する
77          cal_imaginary(charge_node);
78
79          //仮想電荷の電荷量をテキストデータに出力
80          FILE *fp;
81          fp = fopen( "charge_quantity.txt", "w" );
82          if( fp == NULL )
83          {
84                  puts( "ファイルが開けません" );
85                  return 1;
86          }
87          for(int j=0; j<N_image; ++j)
```

```
88          {
89                  fprintf( fp, "%16.16lf", charge_quantity[j]);
90                  fprintf( fp, "\n" );
91          }
92          fclose( fp );
93          return 0;
94  }
95
96  //関数の定義
97  int get_node()
98  {
99          FILE *fp;
100         char lBuf[512],*p;
101
102         if ((fp = fopen("node.txt","r")) == NULL){
103                 fprintf(stderr,"ファイルが開けません\n");
104                 return 1;
105         }
106         for(int m=0; m<N_node; m++ ) {
107                 fgets( lBuf, sizeof( lBuf ), fp );
108
109                 p = strtok( lBuf, "␣␣␣␣␣" );
110
111                 for (int n=0; n<3 && p!=NULL; n++) {
112                         sscanf( p, "%lf", &node[m][n] );
113                         p = strtok( NULL, "␣␣␣␣␣" );
114                 }
115         }
116         fclose(fp);
117         return 0;
118 }
119
120 int get_element()
121 {
122         FILE *fp;
123         char lBuf[512],*p;
124
125         if ((fp = fopen("element.txt","r")) == NULL){
126                 fprintf(stderr,"ファイルが開けません\n");
127                 return 1;
128         }
129         for(int m=0; m<N_ele; m++ ) {
130                 fgets( lBuf, sizeof( lBuf ), fp );
131                 p = strtok( lBuf, "␣␣␣␣␣" );
132                 for (int n=0; n<3 && p!=NULL; n++) {
```

```
133                             sscanf( p, "%d", &element[m][n] );
134                             p = strtok( NULL, "␣␣␣␣␣" );
135                     }
136             }
137             fclose(fp);
138             return 0;
139     }
140
141     void cal_space()
142     {
143             int a,b,c;
144             double l1,l2,l3,s;
145
146             for (int i=0; i<N_ele; i++)
147             {
148                     a=element[i][0]-1;
149                     b=element[i][1]-1;
150                     c=element[i][2]-1;
151
152                     l1=sqrt(pow(node[a][0]-node[b][0],2)+pow(node[a][1]-node[b][1],2)+pow(node
                            [a][2]-node[b][2],2));
153                     l2=sqrt(pow(node[b][0]-node[c][0],2)+pow(node[b][1]-node[c][1],2)+pow(node
                            [b][2]-node[c][2],2));
154                     l3=sqrt(pow(node[a][0]-node[c][0],2)+pow(node[a][1]-node[c][1],2)+pow(node[
                            a][2]-node[c][2],2));
155
156                     s=(l1+l2+l3)/2.0L;
157
158                     space[i] = sqrt(s*(s-l1)*(s-l2)*(s-l3));
159             }
160     }
161
162     void gauss(int m)
163     {
164             if (m==2) {
165                     x[0]=-0.5773502691896257L;
166                     x[1]=0.5773502691896257L;
167                     w[0]=1.0L;
168                     w[1]=1.0L;
169             }
170             if (m==3) {
171                     x[0]=-0.7745966692414834L;
172                     x[1]=0.0L;
173                     x[2]=0.7745966692414834L;
174
```

```
175                 w[0]=0.5555555555555556L;
176                 w[1]=0.8888888888888888L;
177                 w[2]=0.5555555555555556L;
178         }
179         if (m==4) {
180                 x[0]=-0.8611363115940526L;
181                 x[1]=-0.3399810435848563L;
182                 x[2]=0.3399810435848563L;
183                 x[3]=0.8611363115940526L;
184
185                 w[0]=0.3478548451374538L;
186                 w[1]=0.6521451548625463L;
187                 w[2]=0.6521451548625463L;
188                 w[3]=0.3478548451374538L;
189         }
190         if (m==5) {
191                 x[0]=-0.9061798459386641L;
192                 x[1]=-0.5384693101056830L;
193                 x[2]=0.0L;
194                 x[3]=0.5384693101056830L;
195                 x[4]=0.9061798459386641L;
196
197                 w[0]=0.2369268850561892L;
198                 w[1]=0.4786286704993664L;
199                 w[2]=0.5688888888888889L;
200                 w[3]=0.4786286704993664L;
201                 w[4]=0.2369268850561892L;
202         }
203 }
204
205 double cal_Aij1(int i, int k)
206 {
207         int k1,k2,k3;
208         double sum=0;
209         k1=element[k][0]-1;
210         k2=element[k][1]-1;
211         k3=element[k][2]-1;
212
213         for (int a=0; a<NG ; a++)
214         {
215                 for (int b=0; b<NG; b++)
216                 {
217                         sum=sum+0.25*w[a]*w[b]*(1+x[a])*space[k]*((1-x[a])*0.5)/(sqrt(pow
                                (-node[i][0]+node[k1][0]+0.5*(1+x[a])*(node[k2][0]-node[k1
                                ][0])+0.25*(1+x[a])*(1+x[b])*(node[k3][0]-node[k2][0]),2)+pow
```

```
                                (−node[i][1]+node[k1][1]+0.5∗(1+x[a])∗(node[k2][1]−node[k1
                                ][1])+0.25∗(1+x[a])∗(1+x[b])∗(node[k3][1]−node[k2][1]),2)+pow
                                (−node[i][2]+node[k1][2]+0.5∗(1+x[a])∗(node[k2][2]−node[k1
                                ][2])+0.25∗(1+x[a])∗(1+x[b])∗(node[k3][2]−node[k2][2]),2)));
218                     }
219             }
220             return sum;
221     }
222
223     double cal_Aij2(int i, int k)
224     {
225             int k1,k2,k3;
226             double sum=0;
227             k1=element[k][0]−1;
228             k2=element[k][1]−1;
229             k3=element[k][2]−1;
230
231             for (int a=0; a<NG ; a++)
232             {
233                     for (int b=0; b<NG; b++)
234                     {
235                             sum=sum+0.25∗w[a]∗w[b]∗(1+x[a])∗space[k]∗((1+x[a])∗(1−x[b
                                ])∗0.25)/(sqrt(pow(−node[i][0]+node[k1][0]+0.5∗(1+x[a])∗(node[
                                k2][0]−node[k1][0])+0.25∗(1+x[a])∗(1+x[b])∗(node[k3][0]−node[k2
                                ][0]),2)+pow(−node[i][1]+node[k1][1]+0.5∗(1+x[a])∗(node[k2][1]−
                                node[k1][1])+0.25∗(1+x[a])∗(1+x[b])∗(node[k3][1]−node[k2
                                ][1]),2)+pow(−node[i][2]+node[k1][2]+0.5∗(1+x[a])∗(node[k2][2]−
                                node[k1][2])+0.25∗(1+x[a])∗(1+x[b])∗(node[k3][2]−node[k2
                                ][2]),2)));
236                     }
237             }
238             return sum;
239     }
240
241     double cal_Aij3(int i, int k)
242     {
243             int k1,k2,k3;
244             double sum=0;
245             k1=element[k][0]−1;
246             k2=element[k][1]−1;
247             k3=element[k][2]−1;
248
249             for (int a=0; a<NG ; a++)
250             {
251                     for (int b=0; b<NG; b++)
```

```
252                 {
253                         sum=sum+0.25*w[a]*w[b]*(1+x[a])*space[k]*((1+x[a])*(1+x[b
                            ])*0.25)/(sqrt(pow(-node[i][0]+node[k1][0]+0.5*(1+x[a])*(node[
                            k2][0]-node[k1][0])+0.25*(1+x[a])*(1+x[b])*(node[k3][0]-node[k2
                            ][0]),2)+pow(-node[i][1]+node[k1][1]+0.5*(1+x[a])*(node[k2][1]-
                            node[k1][1])+0.25*(1+x[a])*(1+x[b])*(node[k3][1]-node[k2
                            ][1]),2)+pow(-node[i][2]+node[k1][2]+0.5*(1+x[a])*(node[k2][2]-
                            node[k1][2])+0.25*(1+x[a])*(1+x[b])*(node[k3][2]-node[k2
                            ][2]),2)));
254                 }
255         }
256         return sum;
257
258 }
259
260 void DATAIN(const char name[], int nr, int nc,double ma[],double Aij[][N_node])
261 {
262         int i,j;
263         for (i=0; i<nr; i++)
264         {
265                 for (j=0; j<nc; j++)
266                 {
267                         ma[nc*i+j]=Aij[i][j];
268                 }
269         }
270 }
271
272 void DATAIN2(const char name[], int nr, int nc,double ma[],double V_node[])
273 {
274         int i,j;
275         for (i=0; i<nr; i++)
276         {
277                 for (j=0; j<nc; j++)
278                 {
279                         ma[nc*i+j]=V_node[i];
280                 }
281         }
282 }
283
284 int PIVOT(int *num, int nr, int k,double ma[] )
285 {
286         int i;
287         double aa,bb;
288
289         *num=k;
```

```
290         aa=fabs(ma[nr*k+k]);
291         for (i=k+1; i<nr; i++)
292         {
293                 if (fabs(ma[nr*i+k])>aa)
294                 {
295                         *num=i;
296                         aa=fabs(ma[nr*i+k]);
297                 }
298         }
299         if (fabs(aa)<=EPS) return 1;
300         if (*num==k) return 0;
301         for (i=k; i<nr; i++)
302         {
303                 bb=ma[nr*k+i];
304                 ma[nr*k+i]=ma[nr*(*num)+i];
305                 ma[nr*(*num)+i]=bb;
306         }
307         return 0;
308 }
309
310 int GAUSS(int nr,double ma[],double mb[],double mx[])
311 { int i,j,k;
312         int num;
313         double cc;
314
315         for (k=0; k<nr-1; k++)
316         {
317                 if (PIVOT(&num, nr, k, ma)!=0) return 1;
318                 if (num !=k)
319                 {
320                         cc=mb[num]; mb[num]=mb[k]; mb[k]=cc;
321                 }
322                 for (i=k+1; i<nr; i++)
323                 {
324                         cc=ma[nr*i+k]/ma[nr*k+k];
325                         for (j=k+1; j<nr; j++)
326                                 ma[nr*i+j]=ma[nr*i+j]-cc*ma[nr*k+j];
327                         mb[i]=mb[i]-cc*mb[k];
328                 }
329         }
330         for(k=nr-1;k>=0;k--)
331         {
332                 if (fabs(ma[nr*k+k])<=EPS) return 1;
333                 cc=0.0;
334                 for (j=k+1; j<nr; j++)
```

```
335                         cc+=ma[nr*k+j]*mx[j];
336                 mx[k]=(mb[k]-cc)/ma[nr*k+k];
337         }
338         return 0;
339 }
340
341 double get_boundary(double V_node[])
342 {
343         FILE *fp;
344         char lBuf[512],*p;
345
346         if ((fp = fopen("boundary_condition.txt","r")) == NULL){
347                 fprintf(stderr,"ファイルが開けません\n");
348                 return 1;
349         }
350         for(int m=0; m<N_node; m++ ) {
351                 fgets( lBuf, sizeof( lBuf ), fp );
352                 p = strtok( lBuf, "␣␣␣␣␣" );
353                 sscanf( p, "%lf", &V_node[m]);
354                 p = strtok( NULL, "␣␣␣␣␣" );
355         }
356         fclose(fp);
357         return 0;
358 }
359
360 void cal_imaginary(double charge_node[])
361 {
362         for (int k=0; k<N_ele; k++)
363         {
364                 for (int a=0; a<NG; a++)
365                 {
366                         for (int b=0; b<NG; b++)
367                         {
368                                 int i,k1,k2,k3;
369                                 i=k*pow(NG,2)+a*NG+b;
370                                 k1=element[k][0]-1;
371                                 k2=element[k][1]-1;
372                                 k3=element[k][2]-1;
373
374                                 charge_quantity[i]=0.25*space[k]*w[a]*w[b]*(1+x[a])*(
                                        charge_node[k1]*((1-x[a])*0.5)+charge_node[k2]*((1+x[a
                                        ])*(1-x[b])*0.25)+charge_node[k3]*((1+x[a])*(1+x[b
                                        ])*0.25));
375                         }
376                 }
```

```
377             }
378 }
```

ソースコード 9.6: field_3D.cpp（電場を計算するパート）

```
1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  //各種の定数を定義
7  #define NG 5//ガウス積分公式の積分点の数
8  #define N_ele 3364//要素数
9  #define N_node 1740//節点数
10 #define N_image N_ele*25//仮想線電荷の数
11 #define N_cal 30//計算点数
12
13 //グローバル変数
14 double charge_quantity[N_image];//仮想電荷（点電荷）の電荷量を格納
15 int element[N_ele][3];//要素の情報を格納
16 double node[N_node][3];//節点の情報を格納
17 double x[NG],w[NG];//積分点とその重みを格納
18
19 //関数プロトタイプ宣言
20 int get_element();//要素の情報を得る
21 int get_node();//節点の情報を得る
22 void gauss(int m);//ガウス積分公式の積分点、重みを決める
23 double get_calpoint(double calpoint[][3]);//計算点の情報を得る
24 double get_imaginary();//仮想点電荷（点電荷）の電荷量を得る
25 void cal_field(double E[][3],double calpoint[][3], int i);//計算点の電場を計算する
26
27 //メイン関数
28 int main()
29 {
30         //各種情報を得る
31         get_element();
32         get_node();
33         gauss(NG);
34         get_imaginary();
35         double calpoint[N_cal][3];
36         get_calpoint(calpoint);
37
38         //電場の計算結果を書き込むファイルを準備
39         FILE *fp;
40         fp = fopen( "field.txt", "w" );
41         if( fp == NULL )
42         {
43                 puts( "ファイルが開けません" );
44                 return 1;
```

```
45          }
46
47          //電場を計算し、結果をファイルに書き込む
48          double E[N_cal][3]={0.0L};
49          for (int i=0; i<N_cal; i++) {
50                  cal_field( E, calpoint, i);
51                  fprintf( fp, "%16.16lf␣", E[i][0] );
52                  fprintf( fp, "%16.16lf␣", E[i][1] );
53                  fprintf( fp, "%16.16lf␣", E[i][2]);
54                  fprintf( fp, "\n" );
55          }
56          fclose( fp );
57          return 0;
58  }
59
60  //関数の定義
61  int get_node()
62  {
63          FILE *fp;
64          char lBuf[512],*p;
65
66          if ((fp = fopen("node.txt","r")) == NULL){
67                  fprintf(stderr,"ファイルが開けません\n");
68                  return 1;
69          }
70          for(int m=0; m<N_node; m++ ) {
71                  fgets( lBuf, sizeof( lBuf ), fp );
72                  p = strtok( lBuf, "␣␣␣␣␣" );
73                  for (int n=0; n<3 && p!=NULL; n++) {
74                          sscanf( p, "%lf", &node[m][n] );
75                          p = strtok( NULL, "␣␣␣␣␣" );
76                  }
77          }
78          fclose(fp);
79          return 0;
80  }
81
82  int get_element()
83  {
84          FILE *fp;
85          char lBuf[512],*p;
86
87          if ((fp = fopen("element.txt","r")) == NULL){
88                  fprintf(stderr,"ファイルが開けません\n");
89                  return 1;
```

```
90          }
91          for(int m=0; m<N_ele; m++ ) {
92                  fgets( lBuf, sizeof( lBuf ), fp );
93                  p = strtok( lBuf, "␣␣␣␣␣" );
94                  for (int n=0; n<3 && p!=NULL; n++) {
95                          sscanf( p, "%d", &element[m][n] );
96                          p = strtok( NULL, "␣␣␣␣␣" );
97                  }
98          }
99          fclose(fp);
100         return 0;
101 }
102
103 void gauss(int m)
104 {
105         if (m==2) {
106                 x[0]=-0.5773502691896257;
107                 x[1]=0.5773502691896257;
108                 w[0]=1.0;
109                 w[1]=1.0;
110         }
111         if (m==3) {
112                 x[0]=-0.7745966692414834;
113                 x[1]=0.0;
114                 x[2]=0.7745966692414834;
115
116                 w[0]=0.5555555555555556;
117                 w[1]=0.8888888888888888;
118                 w[2]=0.5555555555555556;
119         }
120         if (m==4) {
121                 x[0]=-0.8611363115940526;
122                 x[1]=-0.3399810435848563;
123                 x[2]=0.3399810435848563;
124                 x[3]=0.8611363115940526;
125
126                 w[0]=0.3478548451374538;
127                 w[1]=0.6521451548625463;
128                 w[2]=0.6521451548625463;
129                 w[3]=0.3478548451374538;
130         }
131         if (m==5) {
132                 x[0]=-0.9061798459386641;
133                 x[1]=-0.5384693101056830;
134                 x[2]=0.0;
```

```
135                     x[3]=0.5384693101056830;
136                     x[4]=0.9061798459386641;
137
138                     w[0]=0.2369268850561892;
139                     w[1]=0.4786286704993664;
140                     w[2]=0.5688888888888889;
141                     w[3]=0.4786286704993664;
142                     w[4]=0.2369268850561892;
143             }
144     }
145
146     double get_imaginary()
147     {
148             FILE *fp1;
149             char lBuf_1[512],*p1;
150             if ((fp1 = fopen("charge_quantity.txt","r")) == NULL){
151                     fprintf(stderr,"ファイルが開けません\n");
152                     return 1;
153             }
154             for(int m=0; m<N_image ; m++ ) {
155                     fgets( lBuf_1, sizeof( lBuf_1 ), fp1 );
156                     p1 = strtok( lBuf_1, "␣␣" );
157                     sscanf( p1, "%lf", &charge_quantity[m]);
158                     p1 = strtok( NULL, "␣␣␣␣" );
159             }
160             fclose(fp1);
161             return 0;
162     }
163
164     void cal_field(double E[][3],double calpoint[][3], int i)
165     {
166             for (int k=0; k<N_ele; k++)
167             {
168                     for (int a=0; a<NG; a++)
169                     {
170                             for (int b=0; b<NG; b++)
171                             {
172                                     int t,k1,k2,k3;
173                                     t=k*pow(NG,2)+a*NG+b;
174                                     k1=element[k][0]-1;
175                                     k2=element[k][1]-1;
176                                     k3=element[k][2]-1;
177
178                                     E[i][0] += charge_quantity[t]*(-(-calpoint[i][0]+node[k1
                                            ][0]+0.5L*(1.0L+x[a])*(node[k2][0]-node[k1][0])+0.25L
```

```
                                        *(1.0L+x[a])*(1.0L+x[b])*(node[k3][0]-node[k2][0])))/
                                        pow(pow(-calpoint[i][0]+node[k1][0]+0.5L*(1.0L+x[a])*(
                                        node[k2][0]-node[k1][0])+0.25L*(1.0L+x[a])*(1.0L+x[b
                                        ])*(node[k3][0]-node[k2][0]),2)+pow(-calpoint[i][1]+node
                                        [k1][1]+0.5L*(1.0L+x[a])*(node[k2][1]-node[k1][1])+0.25
                                        L*(1.0L+x[a])*(1.0L+x[b])*(node[k3][1]-node[k2][1]),2)+
                                        pow(-calpoint[i][2]+node[k1][2]+0.5L*(1.0L+x[a])*(node[
                                        k2][2]-node[k1][2])+0.25L*(1.0L+x[a])*(1.0L+x[b])*(
                                        node[k3][2]-node[k2][2]),2),1.5);
179
180                                     E[i][1] += charge_quantity[t]*(-(-calpoint[i][1]+node[k1
                                        ][1]+0.5L*(1.0L+x[a])*(node[k2][1]-node[k1][1])+0.25L
                                        *(1.0L+x[a])*(1.0L+x[b])*(node[k3][1]-node[k2][1])))/
                                        pow(pow(-calpoint[i][0]+node[k1][0]+0.5L*(1.0L+x[a])*(
                                        node[k2][0]-node[k1][0])+0.25L*(1.0L+x[a])*(1.0L+x[b
                                        ])*(node[k3][0]-node[k2][0]),2)+pow(-calpoint[i][1]+node
                                        [k1][1]+0.5L*(1.0L+x[a])*(node[k2][1]-node[k1][1])+0.25
                                        L*(1.0L+x[a])*(1.0L+x[b])*(node[k3][1]-node[k2][1]),2)+
                                        pow(-calpoint[i][2]+node[k1][2]+0.5L*(1.0L+x[a])*(node[
                                        k2][2]-node[k1][2])+0.25L*(1.0L+x[a])*(1.0L+x[b])*(
                                        node[k3][2]-node[k2][2]),2),1.5);
181
182                                     E[i][2] += charge_quantity[t]*(-(-calpoint[i][2]+node[k1
                                        ][2]+0.5L*(1.0L+x[a])*(node[k2][2]-node[k1][2])+0.25L
                                        *(1.0L+x[a])*(1.0L+x[b])*(node[k3][2]-node[k2][2])))/
                                        pow(pow(-calpoint[i][0]+node[k1][0]+0.5L*(1.0L+x[a])*(
                                        node[k2][0]-node[k1][0])+0.25L*(1.0L+x[a])*(1.0L+x[b
                                        ])*(node[k3][0]-node[k2][0]),2)+pow(-calpoint[i][1]+node
                                        [k1][1]+0.5L*(1.0L+x[a])*(node[k2][1]-node[k1][1])+0.25
                                        L*(1.0L+x[a])*(1.0L+x[b])*(node[k3][1]-node[k2][1]),2)+
                                        pow(-calpoint[i][2]+node[k1][2]+0.5L*(1.0L+x[a])*(node[
                                        k2][2]-node[k1][2])+0.25L*(1.0L+x[a])*(1.0L+x[b])*(
                                        node[k3][2]-node[k2][2]),2),1.5);
183                                 }
184                         }
185         }
186 }
187
188 double get_calpoint(double calpoint[][3])
189 {
190         FILE *fp;
191         char lBuf[512],*p;
192
193         if ((fp = fopen("calculation_point.txt","r")) == NULL){
194                 fprintf(stderr,"ファイルが開けません\n");
```

```
195                 return 1;
196         }
197         for(int m=0; m<N_cal; m++ ) {
198                 fgets( lBuf, sizeof( lBuf ), fp );
199                 p = strtok( lBuf, "␣␣␣␣␣" );
200                 for(int n=0;n<3;n++){
201                         sscanf( p, "%lf", &calpoint[m][n]);
202                         p = strtok( NULL, "␣␣␣␣␣" );
203                 }
204         }
205         fclose(fp);
206         return 0;
207 }
```

謝辞

本研究を進めるにあたり、質量分析学の基本から論文執筆にいたるまで、終始ご指導を賜りました指導教官の豊田岐聡先生に深く感謝いたします。また、石原盛男先生、研究員の青木順さんには数値計算について多くの有益な助言をいただきました。その他、質量分析グループ関係者の皆様のお力添えがなければ本研究を進めることはできませんでした。この場をお借りして皆様に深く御礼申し上げます。

平成 24 年 2 月 29 日

安藤弘樹

参考文献

[1] M.Toyoda, M.Ishihara, S.Ymaguchi, H.Ito, T.Matsuo, R.Roll, H.Rosenbauer, *J. Mass. Spectrom.,* 35 (2000) 163.

[2] M.Toyoda, D.Okumura, M.Ishihara and I.Katakuse, *J. Mass. Spectrom.,* 38 (2003), 1125-1142.

[3] W. E. Stephens, *Phys. Rev.,* 69 (1946), 691

[4] K.Hiraoka, S.Fujimaki, S.Kambara, H.Furuya, S.Ozaki, *Rapid Commun. Mass Spectrom.,* 2004, 18, 2323

[5] J.H.J.Dawson and M. Guihaus,*Rapid Commum. Mass Spectrom.,* 3, 155 (1989).

[6] H.Kanou, Master thesis, Osaka University 2010

[7] H.Nagao, H.Kanou, K.Iwamoto, M.Toyoda, *J. Mass. Spectrom.,* 59 (2011)265.

[8] 特許名称：リニアイオントラップ質量分析装置, 発明者：豊田岐聡, 岩本賢一, 木村健二, 出願番号：特願 2007-19693, 出願日：2007 年 1 月 30 日, 出願人:MSI.TOKYO(株)

[9] M.Ishihara, Doctor thesis, Osaka University 1991

[10] 宅間薫, 浜田昌司 「数値電界計算の基礎と応用」 東京電機大学出版局 (2006)

[11] S.Shimma, H.Nagao, J.Aoki, K.Takahashi, S.Miki and M.Toyoda., *Anal.Chem.,* 82 (2010), 8456-8463.

[12] M.Ishihara, M.Toyoda, T.Matsuo,*Int.J.Mass Spectrom.,* 197 (2000) 179-189.

[13] R.B.Cody, J.A.Laramee, J.M.Nilles, H.D.Durst, *JEOL News* 2005, 40 (1), 8

[14] J.C.Schwarrtz, M.W.Senko, and J.E.P.Syka, *J.Am.Soc.Mass Spectrom.,* 13, 659 (2002).

[15] W.C.Wiley and I.H.MClaren, *Rev. Sci. Instrum,.* 26, 1150 (1955).

[16] D.Okumura, K.Kumondai, S.Yamaguchi, M.Toyoda, M.Ishihara and I.Katakuse, *J. Mass. Spectrom. Soc. Jpn.,* 48, 357 (2000).

[17] P.J.Davis, P.Rabinowitz 著 森正武 訳「計算機による数値積分法」科学技術出版社 (1980)

[18] 山崎勝義 「衝突頻度と平均自由行程」 漁火書店 (2008)

[19]「Mersenne Twister Home Page」
http://www.math.sci.hiroshima-u.ac.jp/∼m-mat/MT/mt.html ＞

[20] 久保亮五「大学演習 熱学・統計力学 (修訂第 55 版)」裳華房 (2007) P.467

[21] R.Takai, H.Enokizono, K.Ito, Y.Mizuno, K.Okabe and H.Okamoto., *Japanese Journal of Applied Physics* 45, 6A, (2006), 5332-5343.

[22] Y.Wang, J.Franzen and K.P.Wanczek, *Int.J.Mass Spectrom.Ion Processes,* 124, (1993), 125-144.

[23] A.Drakoudis, M.Sllner, G.Werth, *Int.J.Mass Spectrom.,* 252, (2006), 61-68.

[24] R.Takai, K.Nakayama, W.Saiki, K.Ito and H.Okamoto., *Journal of Physical Society of Japan* 76, (2007).