卒業論文

テーマ：SCRIT におけるイオン分析器の分解能の向上

17cb084r　東條風雅
共同研究者　17cb021b　竹内湧哉
指導教員　栗田和好

1. 目的

理化学研究所に加速器によって生成された不安定原子核を電子ビームポテンシャル内にトラップしておく SCRIT という装置がある（図 1.1）。今回の研究テーマはその不安定原子核を残留ガスなどと識別するためのイオン分析器のアップグレードである。現状、E×B フィルタで軌道をまげて一列に並べられたチャンネルトロンの開口部に入り、電子増幅を行って信号がパルスとして検出する構造になっているが、分解能をより向上していくために基本設計から見直していくことを目的としている。

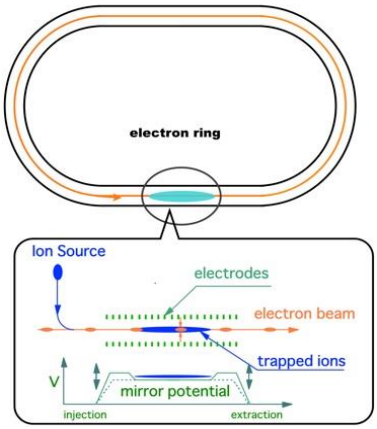

図 1.1：SCRIT のイメージ図

2. 原理

2.1 SCRIT

SCRIT からイオン検出器までの流れは図 2.1.1 のようになっていて、以下の手順でイオンを分別し、検出している。

① SCRIT のポテンシャル壁のうち、片側の山がなくなる。
② SCRIT のポテンシャルから 0keV まで下ることでイオンが加速される。
③ E×B フィルタ（図 2.1.2）で速度、質量、電荷によって分けられる。
④ 検出器に到達し、イオンを検出する。

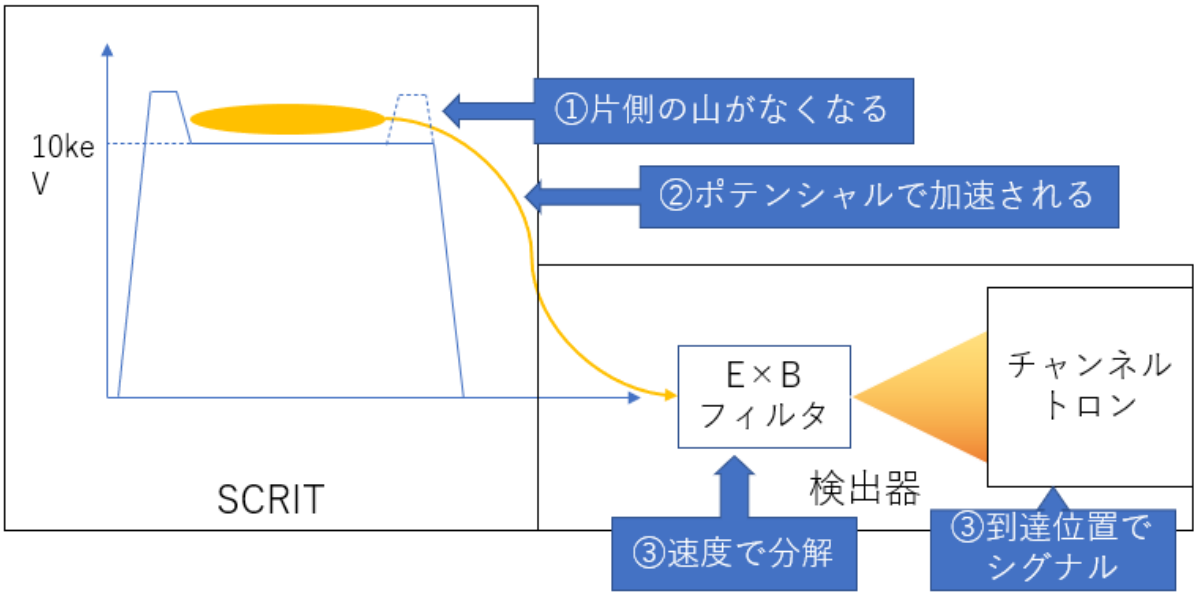

図 2.1.1：SCRIT からイオン検出器まで

SCRIT から来たイオンを分別する E×B フィルタを図 2.1.2 に示す。E×B フィルタでは、電場による力と、磁場によるローレンツ力によって入射方向と垂直な方向に加速させることでイオンを分別している。電場と磁場がかかっているときのイオンの運動方程式は、

$$\frac{d^2r}{dt^2} = \frac{q}{m}(E + v \times B) \qquad \text{式(2.1.1)}$$

であるから、図 2.1.2 の右図では陽イオンに対して、電場によって右方向に、磁場によって左方向に曲げられる。

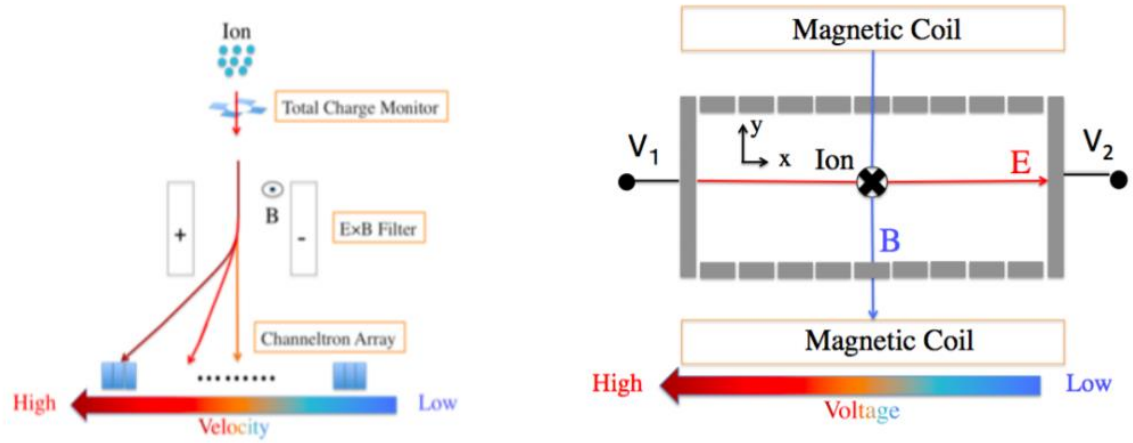

図 2.1.2：E×B フィルタ

現在、イオン検出器としてチャンネルトロン（図 2.1.3）を使用している。イオンの入射方向に対して垂直に 43 個のチャンネルトロンの開口部が 5mm 幅（有感領域 4mm）で並んでいるため分解能は 5mm である。

E×B フィルタで分別されたイオンがチャンネルトロンの別々の開口部に入り、増倍された 2 次電子が信号として検出されることで SCRIT から取り出したイオンの種類の分布を知ることができる。

図 2.1.3：チャンネルトロン

2.2 MCP（マイクロチャンネルプレート）

MCP は内径 6~20 μm の多数のチャンネルで形成され，それぞれが独立の電子増倍管として動作する。MCP は使用時に入口と出口の間に±2000V 程度のバイアス電圧を印加する。符号は検出するイオンの符号に依存する。電子，イオン等がチャンネルに入射すると，チャンネルの内壁に衝突し 2 次電子が放出され，これがバイアス電圧により加速されるため反対側の内壁に衝突し再び 2 次電子を放出していく。この現象が繰り返されることにより，電子が指数関数的に増倍されていく。増幅率は入射電子や二次電子がチャンネルの壁に何回あたるかに依存するため，MCP 入口への入射角度によることとなる。MCP 1 枚の増幅率は 10^3~10^4 であるが，MCP を 2 枚重ね合わせて 10^6~10^7 の増幅率を得たものがよく使われる。

図 2.2.1：MCP

MCP からの信号は MCP 後方に読み出し電極を設置することにより取得する。読み出し電極と MCP 後面の間に 100V 程度印加する。MCP から放出される電子が読み出し電極に引き込まれ，そのパルスを計測することでイオンの検出が行える。

3. 方針

3.1 シミュレーション

今回、基本設計から見直していくためにシミュレーションで十分な分解能を得られる条件を調べていき、設計、実験という方針で進めていく（図 3.1)。イオン軌道計算の一般的なシミュレーションソフトとして simion がよく使われているが、自分で作ったプログラムのほうがより物理を深く学べること、このコロナ下で研究室にも行きづらい状況を考慮して共同研究しやすいソフトを開発してしまったほうがいいと判断しシミュレーションソフトの開発から始めた。

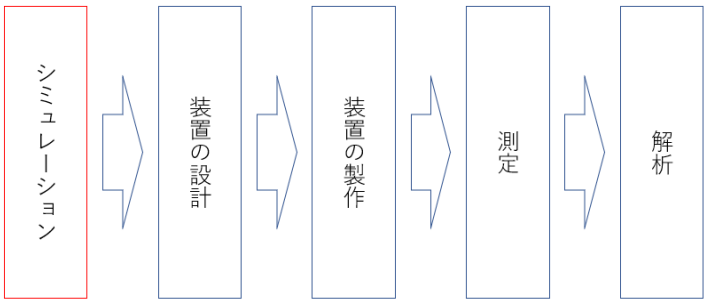

図 3.1：研究のロードマップ

3.2 クロストークの実測

今回の研究では先行研究の懸念点としてクロストークがあげられていた。この問題をクリアするために、まず作成した電極でのクロストークがどの程度あるのかの検証を行い、その結果をもとに MCP での実験に移っていく。実験には 2 ㎜幅、1 ㎜幅、0.5 ㎜幅の 3 種類の電極を使用した。

クロストークの実測については、電極にパルス信号を送った時の 1 つ隣の電極と 2 つ隣の電極の振る舞いをオシロスコープで観察することで測定する。また、配線同士のクロストークも同様の方法で測定する。

4. 進捗

4.1 シミュレーション

4.1.1 シミュレーションの方針

シミュレーションソフトの開発について、言語は python 開発環境として github を利用した。方針としては図 4.1.1 のフローチャートに沿って開発していく。

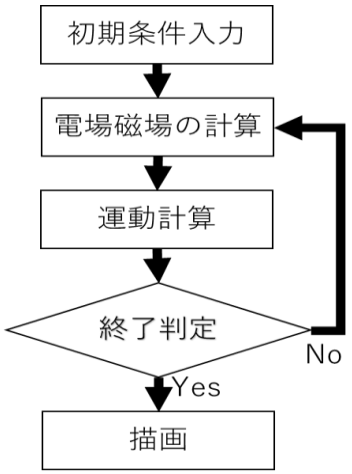

図 4.1.1：システムのフローチャー

4.1.2 各工程のアルゴリズム

①電場、磁場の計算

現状理想的な並行板コンデンサの一様電場を仮定して任意の立方体内のみに一様場を生成するものでシミュレーションしている。

②運動計算

オイラー法、ホルン法、ルンゲクッタ法を比べてみる。図 4.1.2 を見るとわかるようにグラフを見ると、ルンゲクッタ法が最も精度が良い。式はオイラー法が最も簡単であり、実装も容易であると考えられるが、実際のコードではルンゲクッタ法も 4 次までなら複雑にはならない。これらを考慮すると、4 次のルンゲクッタ法が最も適していると判断し、運動方程式を解くアルゴリ

ズムとして使用する。

オイラー法の公式:

$\frac{dy}{dx} = f(x,y)$ ならば、

$$y_{n+1} = y_n + f(x_n, y_n)\Delta h$$

ホルン法の公式:

$\frac{dy}{dx} = f(x,y)$ ならば、

$$k_1 = \Delta h \cdot f(x_n, y_n)$$

$$k_2 = \Delta h \cdot f(x_n + \Delta h, y_n + k_1)$$

$$y_{n+1} = y_n + (k_1 + k_2)/2$$

4次のルンゲクッタ法の公式：

$\frac{dy}{dx} = f(x,y)$ならば、

$$k_1 = f(y_n, x_n)$$

$$k_2 = f\left(y_n + k_1\frac{\Delta h}{2}, x_n + \frac{\Delta h}{2}\right)$$

$$k_3 = f\left(y_n + k_2\frac{\Delta h}{2}, x_n + \frac{\Delta h}{2}\right)$$

$$k_4 = f(y_n + k_3\Delta h, x_n + \Delta h)$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\Delta h$$

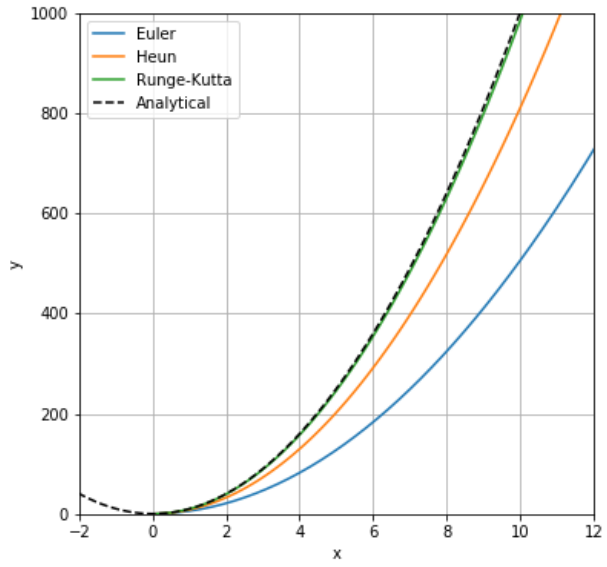

図 4.1.2：各計算方法の公式と精度のグラフ（$f(x,y) = y' = \frac{2y}{x}$、$\Delta h = 0.1$）

③終了判定

指定領域内から出たときを終了条件としている。図 4.1.1 の繰り返し文として while 文を使っていて、シミュレーションする運動によっては無限ループになる可能性が考えられるため、ループ回数の最大値を設定できるようにもしている。

4.1.3 開発

①初期条件の入力

初期条件を保持するファイルとして、とりあえず csv ファイルとして保存した。開発していくうちに電場と磁場を計算するときの初期条件も同じファイルに保存しておきたいと考えたが、csv ファイルでは配列を保存するとわかりづらくなってしまう。そこで、ファイル自体がディクショナリ型で保持できる json ファイルに変更した。図 4.1.3 は実際の json ファイルである。

```
{

  "EFieldplams":{
          "startPos":{"x":-1000,"y":-1000,"z":-1000},
          "endPos":{"x":1000,"y":1000,"z":1000},
          "vector":{"x":0.0,"y":0.0,"z":0.0}
        },

  "BFieldplams":{
          "startPos":{"x":-1000,"y":-1000,"z":-1000},
          "endPos":{"x":1000,"y":1000,"z":1000},
          "vector":{"x":0.0,"y":0.0505,"z":0.0}
        },

  "particlePlams":[
                  {"name":"Al:vec10000","kind":"Al","val":3,"pos":[0,0,0],"vec":[10000,0,0]}
                  ]

}
```


図 4.1.3：実際の json ファイルのコード

②電場磁場の計算

最初は簡単に電場と磁場を設定できるように、指定した範囲内に一様電場、一様磁場をかけるという方法をとった。図 4.1.4 はイメージ図である。startpos と endpos の座標を与えることでその間の直方体内に設定した一様電場と一様磁場がかかるようにコードを書いた（図 4.1.5）。

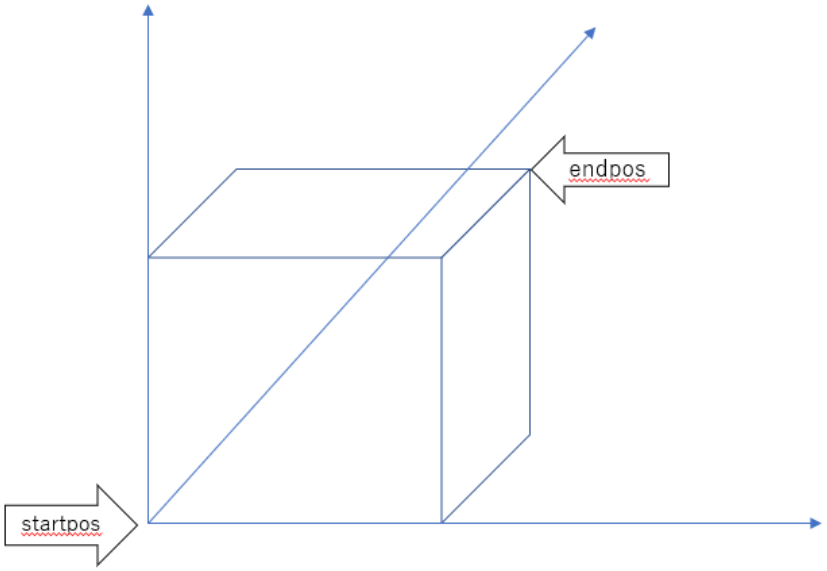

図 4.1.4：電場磁場のイメージ図

```python
#とりあえず、電荷を気にせず特定の座標範囲に並行電場を生成する関数を作ってみる。
class SampleFunc():
    def __init__(self,fieldplams):
        #print("setdata" + str(s_pos) + str(e_pos) + str(vector))
        self.s_pos = fieldplams["startPos"]
        self.e_pos = fieldplams["endPos"]
        self.vector = fieldplams["vector"]
        self.zero_vector = {"x" : 0,"y" : 0 , "z" : 0}
    def VectorField(self,position):
        flag = True
        for ax in position:
            if self.s_pos[ax] <= position[ax] <= self.e_pos[ax]:
                 pass
            else:
                flag = False
        if flag:
            return self.vector
        else:
            return self.zero_vector
```

図 4.1.5：電場磁場の実際のコード

③運動方程式を解く

運動方程式を解く方法として、4 次のルンゲクッタ法を使った。図 4.1.2 の公式は 1 つの微分方程式を解くときの公式であるが、運動方程式は 2 つの連立した微分方程式を解くため、図 4.1.2 よりも複雑な計算をしている。実際の計算式を図 4.1.6 に、コードを図 4.1.7 に示す。

$\frac{dv}{dt} = \boldsymbol{f}(\boldsymbol{v},\boldsymbol{r},t)$, $\frac{dr}{dt} = \boldsymbol{V}(\boldsymbol{v},\boldsymbol{r},t) = \boldsymbol{v}$のとき

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \frac{1}{6}(k_{v1} + 2k_{v2} + 2k_{v3} + k_{v4})\Delta t$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \frac{1}{6}(k_{r1} + 2k_{r2} + 2k_{r3} + k_{r4})\Delta t$$

$$k_{r1} = \boldsymbol{v}_n \qquad k_{v1} = f(\boldsymbol{v}_n, \boldsymbol{r}_n, t_n)$$

$$k_{r2} = \boldsymbol{v}_n + k_{v1}\frac{\Delta t}{2} \qquad k_{v2} = f\left(\boldsymbol{v}_n + k_{v1}\frac{\Delta t}{2}, \boldsymbol{r}_n + k_{r1}\frac{\Delta t}{2}, t_n + \frac{\Delta t}{2}\right)$$

$$k_{r3} = \boldsymbol{v}_n + k_{v2}\frac{\Delta t}{2} \qquad k_{v3} = f\left(\boldsymbol{v}_n + k_{v2}\frac{\Delta t}{2}, \boldsymbol{r}_n + k_{r2}\frac{\Delta t}{2}, t_n + \frac{\Delta t}{2}\right)$$

$$k_{r4} = \boldsymbol{v}_n + k_{v3}\Delta t \qquad k_{v4} = f(\boldsymbol{v}_n + k_{v3}\Delta t, \boldsymbol{r}_n + k_{r3}\Delta t, t_n + \Delta t)$$

図 4.1.6：運動方程式を解くときの 4 次のルンゲクッタ法の計算式

```
k1_x = vec[i]
k1_v = Force(t, pos[i], vec[i], E, B, q, m)
k2_x = vec[i] + k1_v*tmp_dt/2
k2_v = Force(t + tmp_dt/2, pos[i] + k1_x*tmp_dt/2, vec[i] + k1_v*tmp_dt/2, E, B, q, m)
k3_x = vec[i] + k2_v*tmp_dt/2
k3_v = Force(t + tmp_dt/2, pos[i] + k2_x*tmp_dt/2, vec[i] + k2_v*tmp_dt/2, E, B, q, m)
k4_x = vec[i] + k3_v*tmp_dt
k4_v = Force(t + tmp_dt, pos[i] + k3_x*tmp_dt, vec[i] + k3_v*tmp_dt, E, B, q, m)

vec_tem = vec[i] + tmp_dt*(k1_v + 2*k2_v + 2*k3_v + k4_v)/6
vec = np.append(vec, np.array([vec_tem]), axis=0)
pos_tem = pos[i] + tmp_dt*(k1_x + 2*k2_x + 2*k3_x + k4_x)/6
pos = np.append(pos, np.array([pos_tem]), axis=0)
```

図 4.1.7： 4 次のルンゲクッタ法の実際のコード

4.1.4 動作テスト

これらのシステムが想定通りに機能していることの確認のために、運動計算と電場、磁場でそれぞれテストを行った。方針としては、電場のテストと磁場のテストそれぞれについて解析的に求めた値とプログラムから求めた値が一致していることを確認する。

① 電場のテスト

z 軸方向に電場をかけたときに、$z_1 = 200[\mathrm{mm}]$を超えるまでの時間Δt を比較する。このとき、運動方程式は、

$$\frac{d^2z}{dt^2} = \frac{q}{m}E \qquad \text{式(4.1.1)}$$

であるから、これを解析的に解くと、

$$\Delta t = \sqrt{\frac{2mz_1}{qE}} \qquad \text{式(4.1.2)}$$

である。プログラムでは$z_1 = 200[\mathrm{mm}]$を超えることを終了条件としてシミュレーションする。設定と初期条件は右のとおりである。

解析的に解いた結果は、

$$\Delta t = 8.64265350336229 \times 10^{-6}\ [\mathrm{s}]$$

で、プログラムから求めた値は、

$$\Delta t = 8.643 \times 10^{-6}[\mathrm{s}]$$

であり、その差は

$$3.4649663771069237 \times 10^{-10}[\mathrm{s}]$$

であった。したがって、誤差は約 0.004%でこの値は十分に小さいとし、プログラムは正しく動作していると判断した。実際の実行画面を図 4.1.8 に示す。

設定と初期条件

電場：
(x,y,z) = (-0.1,-0.1,-0.1)[m]から
(x,y,z) = (1.0,1.0,1.0)[m]の立方体内に
一様電場$\boldsymbol{E} = (0,0,500)[\frac{\mathrm{V}}{\mathrm{m}}]$をかける

入射イオン：
電荷+3のアルミニウムイオンAl^{+3}

初期位置：
$$\boldsymbol{r_0} = (0,0,0)[\mathrm{m}]$$

初速度：
$$\boldsymbol{v_0} = (0,0,0)[\frac{\mathrm{m}}{\mathrm{s}}]$$

刻み幅：
$$dt = 10^{-9}[\mathrm{s}]$$

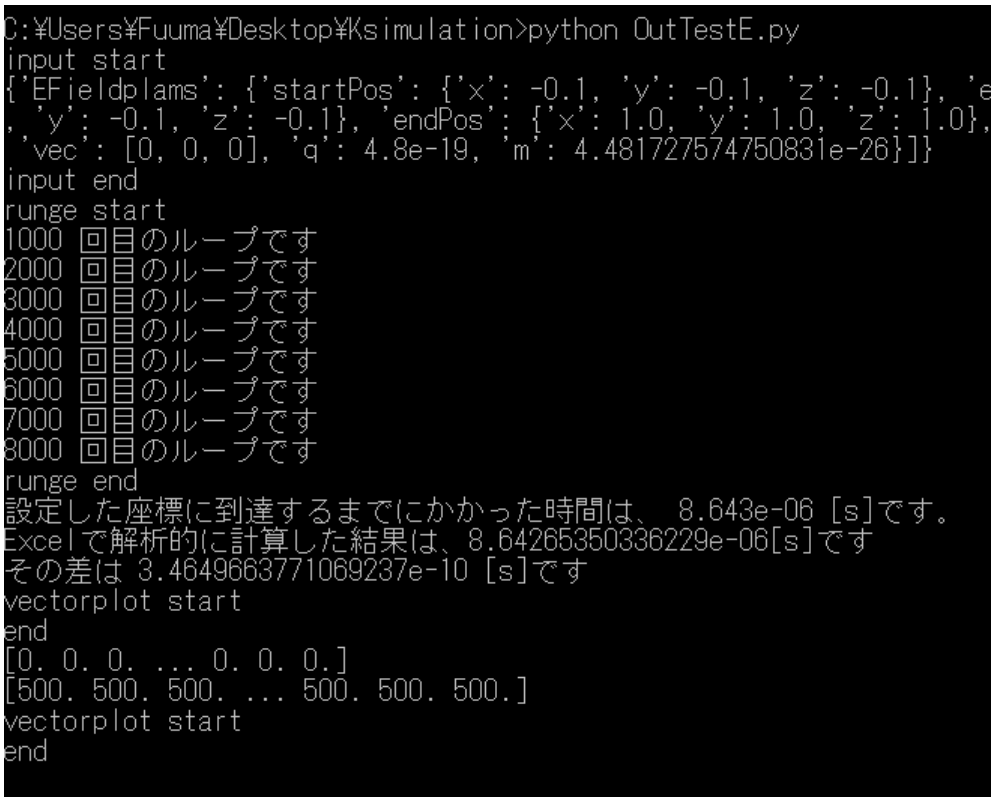

図 4.1.8： 電場のテストの実行画面

② 磁場のテスト

y 軸方向に磁場をかけて、x 軸方向に初速を与えることで、x-z 平面内で円運動をさせる。このときの円の半径を比較する。このとき、運動方程式は、

$$\frac{d^2\boldsymbol{r}}{dt^2} = \frac{q}{m}(\boldsymbol{v} \times \boldsymbol{B}) \qquad \text{式(4.1.3)}$$

であるから、これを解析的に解くと、

$$\mathrm{R} = \frac{mv_0}{qB} \qquad \text{式(4.1.4)}$$

である。プログラムでは以下の手順で円の半径を求める。

【プログラムでの円の半径の求め方】

ⅰ.4 次のルンゲクッタ法で運動方程式を解く

$$r = [(x_0, y_0), (x_1, y_1), \dots (x_n, y_n)] \qquad \text{式(4.1.5)}$$

ⅱ.ルンゲクッタ法の終了条件は 2 周を超えたとき

ⅲ.座標の平均をとって、円の中心(x_c, y_c)を求める

$$x_c = \frac{1}{n+1}\sum_{i=0}^{n} x_i \ , y_c = \frac{1}{n+1}\sum_{i=0}^{n} y_i \qquad \text{式(4.1.6)}$$

ⅳ.各点と円の中心の距離を計算し、その平均値を円の半径とする

$$R = \frac{1}{n+1}\sum_{i=0}^{n}\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \qquad \text{式(4.1.7)}$$

設定と初期条件は右のとおりである。

解析的に解いた結果は、

$$R = 18.4889751433615[\mathrm{mm}]$$

で、プログラムから求めた値は、

$$R = 18.488975114030275[\mathrm{mm}]$$

であり、その差は

$$2.933122544845901 \times 10^{-8}[\mathrm{mm}]$$

であった。したがって、誤差は0.16×10^{-6}[%]未満でこの値は十分に小さいとし、プログラムは正しく動作していると判断した。実際の実行画面を図 4.1.9 に示す。

設定と初期条件

磁場：
(x,y,z) = (-1000,-1000,-1000)[m]から(x,y,z) = (1000,1000,1000)[m]の立方体内に一様磁場$\boldsymbol{B} = (0,0.0505,0)$[T]をかける

入射イオン：
電荷+3のアルミニウムイオンAl^{+3}

初期位置：

$$\boldsymbol{r_0} = (0,0,0)[m]$$

初速度：

$$\boldsymbol{v_0} = (10000,0,0)[\frac{m}{s}]$$

刻み幅：

$$dt = 10^{-9}[\mathrm{s}]$$

```
runge start
1000 回目のループです
2000 回目のループです
3000 回目のループです
4000 回目のループです
5000 回目のループです
6000 回目のループです
7000 回目のループです
8000 回目のループです
9000 回目のループです
10000 回目のループです
11000 回目のループです
12000 回目のループです
13000 回目のループです
14000 回目のループです
15000 回目のループです
16000 回目のループです
17000 回目のループです
18000 回目のループです
19000 回目のループです
20000 回目のループです
21000 回目のループです
22000 回目のループです
23000 回目のループです
runge end
中心の座標は、[X, Z] =  1.5776563472040913e-08 18.48812481213331 です
平均値から求めた半径は、 18.488975114030275 です
分散は、 3.6154822528877l5e-07 です
Excelで解析的に求めた半径は18.4889751433615[mm]で、その差は -2.933122544845901e-08 [mm]です
vectorplot start
end
[]
[]
vectorplot start
end
```

図 4.1.9： 磁場のテストの実行画面

4.1.5 価数が違うイオンのシミュレーション

シミュレーションのテストとして、1 価から 20 価までの ^{132}Sn イオンを判別できるか確認した。まず、装置の大きさを仕様書から読み取り、各軸の大きさと入射イオンの初期位置を設定する。また、SCRIT 内のイオンのポテンシャルは 10keV であるから、この値と運動エネルギーの公式

$$10[\mathrm{keV}] = 1.6 \times 10^{-19} \times 10[\mathrm{J}] = \frac{1}{2} m v^2 \qquad \text{式(4.1.8)}$$

から初速度を決定した。次に、E×B フィルタ内に各イオンが最もセパレートできるような電場と磁場をかける。このとき、E×B フィルタの最大電場と最大磁場を超えないように調整する。結果を図 4.1.10 に示す。

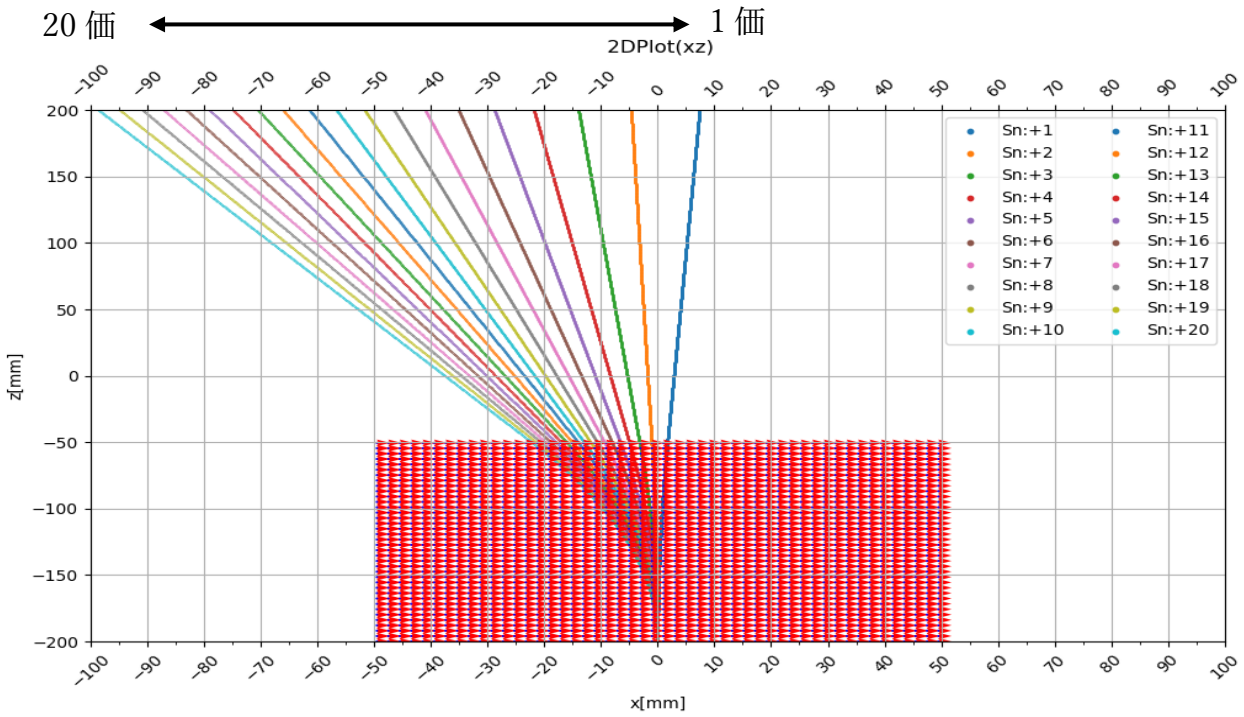

図 4.1.10：Sn を入射したときのシミュレーション結

4.2 MCP 読み出し電極作成

シミュレーションの結果から 20 価までのイオンを完全に分離するためには位置分解能が 2 ㎜以下である必要がある。したがって、先行研究で使用していた 2mm 幅でも十分な性能が出せる可能性があるがより細かい電極ではどうなのかを調べるために 1mm 幅、0.5mm 幅の電極の作成を行った。作成には、P 板.com 社の基板製造サービスを利用したが、電極間の最小が 0.075mm だったため、隙間含め幅 1.075mm、有感領域 1mm の電極と幅 0.575m、有感領域 0.5mm の電極を作成した。

設計した基板の全体イメージを図 4.2.1 に示す。基板は直径 75.0 ㎜、両面基板で製作し，レジストを塗らずにパターンの銅箔(厚さ $18\,\mu$m)を剥き出しにした状態にすることで電極とした。電極は左側がパターン幅 1.0 ㎜パターン間隙 0.075 ㎜の電極 9 本、右側がパターン幅 0.5 ㎜パターン間隙 0.075 ㎜の電極 9 本である。また、各電極について、グラウンドを共有できるようにした。

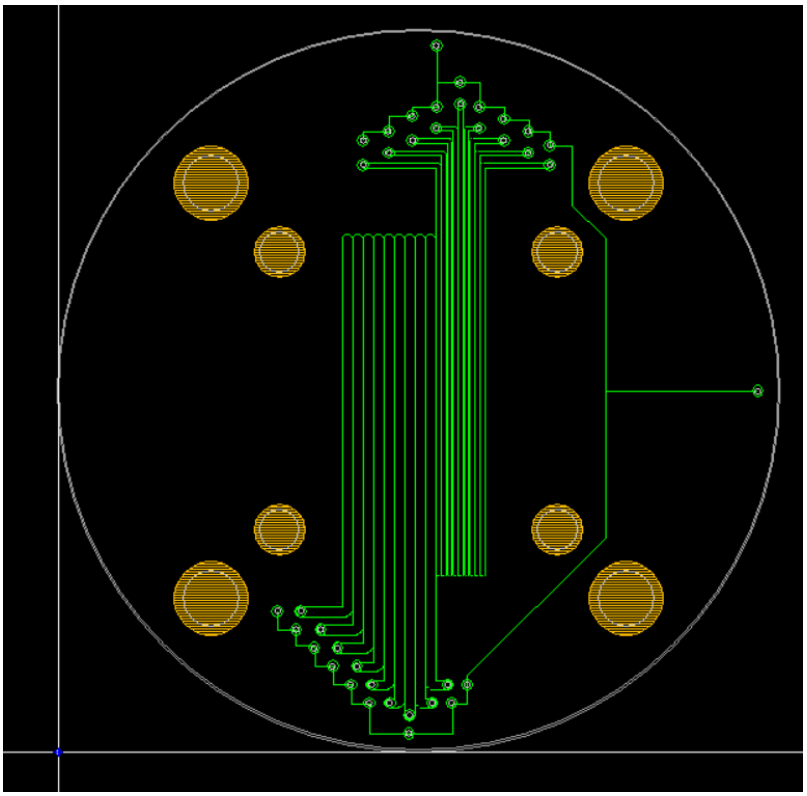

図 4.2.1：設計した基板の全体イメージ

実際に完成した基板を図 4.2.2 に示す。

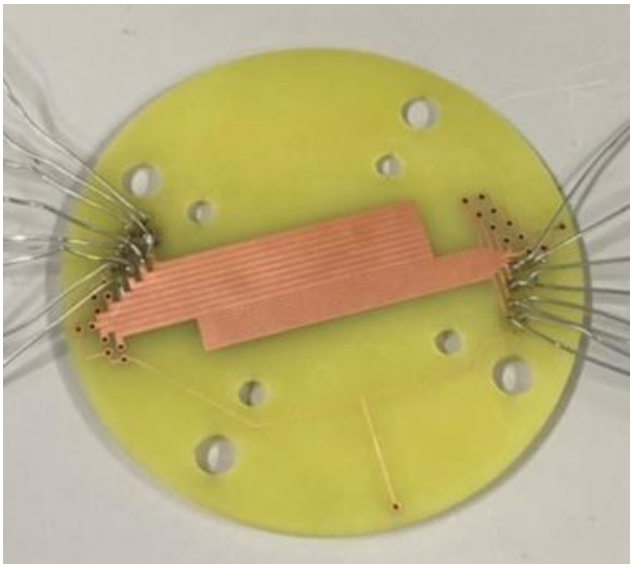

図 4.2.2：完成した基板

4.3 クロストークの実測

MCP からくるパルスの波形を図 4.3.1 に示す。図 4.3.1 の波高バリエーションは最大値を 100%としたときに 75%～100%の値をとる。また、立ち上がり時間は図 4.3.2（図 4.3.1 の拡大図）赤線間の 0V~ピークのうち、10%～90%を読み取って 0.6ns だった。

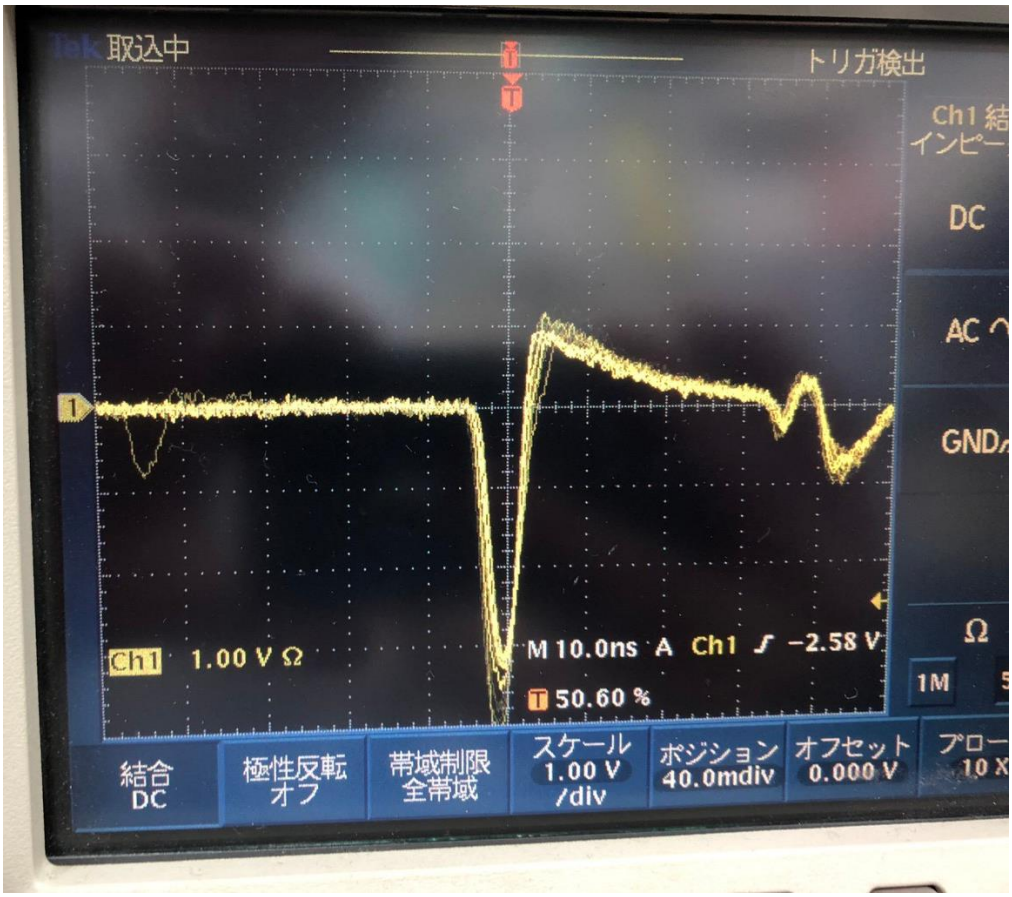

図 4.3.1： MCP からくるパルス波形

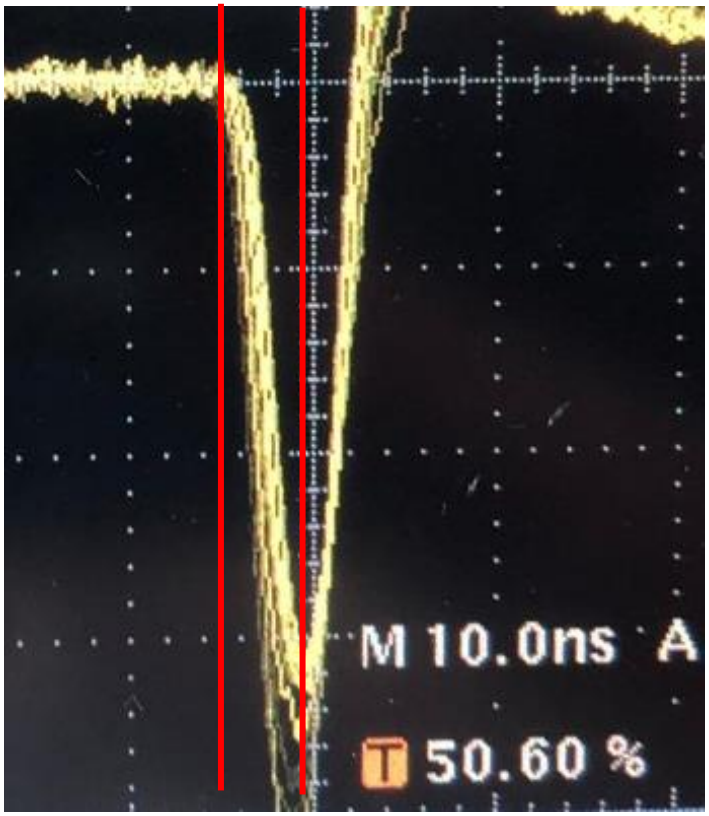

図 4.3.2： MCP からくるパルス波形（拡大図）

クロストークの測定結果を表 4.3.1 に示す。クロストークの測定では図 4.2.1 の左端の電極にパルスを入れ、その一つ隣の電極のパルスを測定した。立ち上がり時間の逆数とインプット電圧に対するクロストークの割合の関係（図 4.3.3 と図 4.3.4）を直線で近似する。MCP からくるパルスの立ち上がり時間は 0.6ns であるから、各電極についてインプットに対するクロストークの割合求めると、0.5mm では 33%、1mm では 38%である。MCP からくるパルスの波高の最小値は最大値に対して 75%程度であるため、閾値を適当な値（最大値の 50%など）に設定すればクロストークをカウントせずに MCP からのパルスのみをカウントすることができる。

表 4.3.1： クロストークの実測

立ち上がり[ns]	1mm	0.5mm
2.58	0.235	0.235
13.00	0.198	0.241
79.20	0.193	0.193

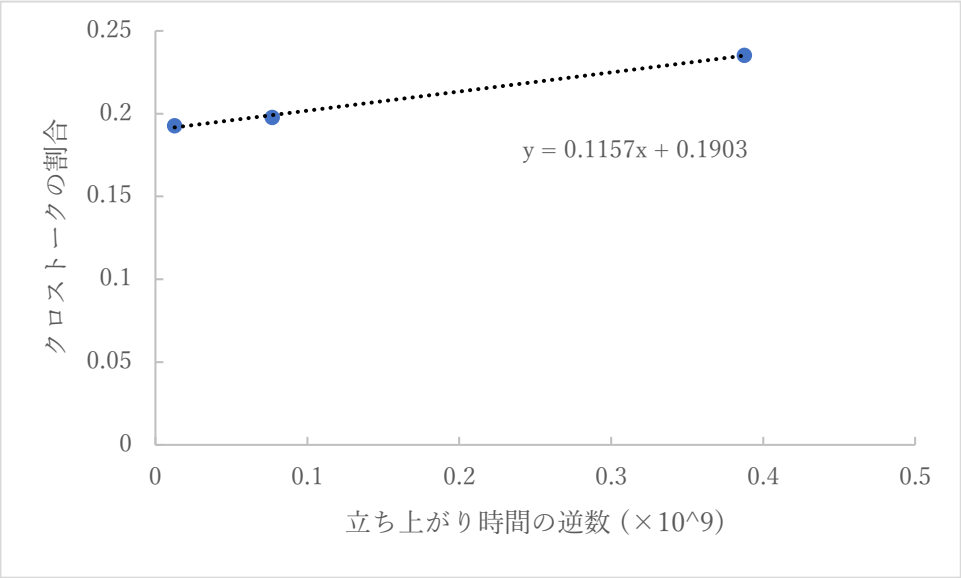

図 4.3.3 ： クロストークの実測（ 1.0 ㎜ ）

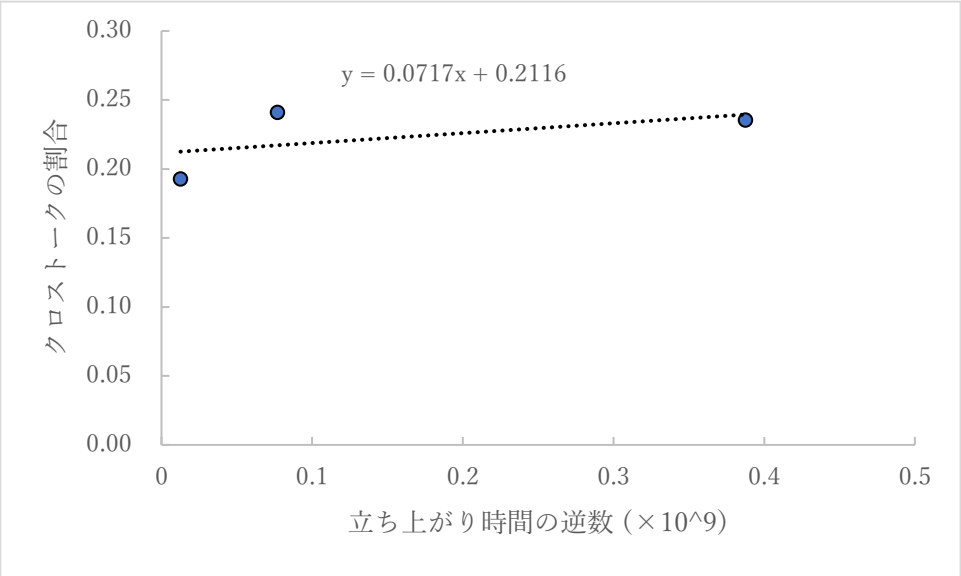

図 4.3.4 ： クロストークの実測（ 0.5㎜ ）

5. 展望

今回の研究によって、シミュレーションによって 2mm 以下の幅で十分な性能を出すことが可能であると予想を立てることができたほか、電極の幅によってクロストークの影響がどの程度でることがわかることで、今後のイオン分析器の性能向上に一つ可能性を示すことができたと考えている。この研究から実際に、分解が可能であることを実験で示し SCRIT に実装されることを期待している。

6. 引用

http://apollo.lns.tohoku.ac.jp/scrit/SCRIT_Sendai_J/Physics.html
http://www.ppl.k.u-tokyo.ac.jp/saito/memo_simion.html
https://automatic-browsing.com/2020/02/28
..¥..¥卒研¥2019 年度卒論栗田研榊原諒.pdf

7. プログラムの保存先

https://github.com/Takeuchi-yuya/Ksimulation

電場のテスト…OutTestE.py
磁場のテスト…OutTestB.py
^{132}Sn イオンのシミュレーション…main.py
初期条件の入力関数…functions/InPut.py
描画…functions/OutPut.py
電場の計算（ラプラス方程式）…functions/makeEfieldMap.c
その他の関数…functions/subtool.py
初期条件ファイル…functions/data