ゼミ レポート 03

山田朔也

2022 年 5 月 23 日

1　本レポートについて

5 月 17 日に行われたゼミにて出題された課題に対するレポートとなっている。課題の内容は、llg 方程式によって導かれる原子磁気モーメントの反転をシュミレーションすることだ。

2　問題 1

2.1　原理

2.1.1　磁気モーメント

磁気モーメントは磁極の対を表す物理量となっている。このとき、磁極の強さを q、磁極の距離を l とし、磁気モーメント自体は $|\vec{M}| = q\vec{l}$ と表す。ここで、距離 l は磁極の値が負から正の方向に向かうベクトルとするため、磁気モーメントも負から正への方向へのベクトル量となる。

また、本レポートでは CGS 単位系を用いるため、$|\vec{M}|$ の単位は emu となる。

2.1.2　原子磁気モーメント

原子磁気モーメントは、原子単体が持つ磁気モーメントのことを指す。また、磁界中に原子磁気モーメントを置くと、原子磁気モーメントは磁界を軸に歳差運動をしつつ磁界の方向を向いていく。

この運動は Landau-Lifshitz-Gilbert 方程式 (LLG 方程式) という微分方程式で表される。

2.1.3　LLG 方程式

LLG 方程式は以下の式 1 で表される。

$$\dot{\vec{M}} = -|\gamma|(\vec{M} \times \vec{H}) + \frac{\alpha}{M}(\vec{M} \times \dot{\vec{M}}) \tag{1}$$

各変数、$\vec{M}$ は原子磁気モーメント、$\vec{H}$ は原子磁気モーメントに加わる実効磁界、γ は磁気回転比、α は損失定数、M は原子磁気モーメントの大きさ ($|\vec{M}|$) となっている。

ここで、原子磁気モーメントを $\vec{M} = M\vec{m}$ とおくと、式 1 は

$$\dot{\vec{m}} = -|\gamma|\vec{m} \times \vec{H} + \alpha\vec{m} \times \dot{\vec{m}} \tag{2}$$

と表せる。また、この式 2 の両辺に $\vec{m}$ を掛けて更に整理すると

$$\vec{m} \times \dot{\vec{m}} = -|\gamma|\vec{m} \times (\vec{m} \times \vec{H}) + \alpha\vec{m} \times (\vec{m} \times \dot{\vec{m}}) \tag{3}$$

$$\Rightarrow \dot{\vec{m}} = \frac{-|\gamma|}{1+\alpha^2}(\vec{m} \times \vec{H} + \alpha(c\vec{m} - \vec{H})) \tag{4}$$

となる。ただし、$c = \vec{m} \cdot \vec{H}$ とする。

また、この式 4 を 3 次元空間上で m_x, m_y, m_z それぞれの式に変えると

$$\begin{aligned}
\dot{m_x} &= \frac{-|\gamma|}{1+\alpha^2}(m_y H_z - m_z H_y + \alpha(cm_x - H_x)) \\
\dot{m_y} &= \frac{-|\gamma|}{1+\alpha^2}(m_z H_x - m_x H_z + \alpha(cm_y - H_y)) \\
\dot{m_z} &= \frac{-|\gamma|}{1+\alpha^2}(m_x H_y - m_y H_x + \alpha(cm_z - H_z))
\end{aligned}$$

となる。ただし

$$\vec{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix},\ \vec{H} = \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix},\ c = m_x H_x + m_y H_y + m_z H_z \tag{5}$$

とする。

2.1.4 計算モデル

本問題では、第二回のレポートに示した 4 次のルンゲクッタ法を用いる。

2.2 小問 1

2.2.1 問題内容

この小問の問題内容は、損失定数を 1 とし、外部磁界 $(0.001, 0, -1)$Oe を加えた場合の原子磁気モーメントの反転シュミレーションを行い、結果を 2 次元および 3 次元のグラフで示す。但し時間刻みは 1ps、磁気回転比は 1.76×10^7rad/(Oe · s) なお、計算の都度原子磁気モーメントの再規格化を行うこと。というものとなっている。

2.2.2 使用したプログラム

まずは以下に作成したプログラムを記載する。

Listing 1: 4 次のルンゲクッタ法で解を得るプログラム

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <stdbool.h>
4   #include <string.h>
5   #include <math.h>
6   #include "solver_RK4.h"
7
8
9
10  int main(int argc, char *argv[]) {
11      double gamma = 0;
12      double alpha = 0.0;
13      double dt = 0.0;
```

```
14              double ku = 0;
15              double m[3];
16              double H[3];
17              double loops = 0;
18              int plots = 0;
19              init(m, H, &dt, &alpha, &gamma, &ku, &loops, &plots);
20
21              int t = 0;
22              for (t = 0; t <= loops; t++) {
23                      RK4(m, H, dt, alpha, gamma, ku);
24                      if (t % plots == 0) {
25                              if (strcmp(argv[1], "3") == 0) {
26                                      printout3(m, t);
27                              } else if (strcmp(argv[1], "2") == 0) {
28                                      printout2(m, t);
29                              }
30                      }
31              }
32              return 0;
33      }
34
35      // ----------------------------------------
36      // 初期設定
37      // ----------------------------------------
38      int init(double* m, double* H, double* dt, double* alpha, double* gamma,
            double* ku, double* loops, int* plots) {
39              FILE *fp;
40              const int n = 256;
41              char fname[] = "init.data";
42              char line[n];
43              char str[16];
44
45              fp = fopen(fname, "r");
46              if (fp == NULL) {
47                      printf("init.data␣not␣open!\n");
48                      exit(1);
49              }
50
51              // 中身の取得
52              while(fgets(line, n, fp) != NULL) {
53                      switch (line[0]){
54                      case '#':
55                              break;
56                      case 'm':
57                              sscanf(line, "%s␣%lf␣%lf␣%lf", str, &m[0], &m[1], &m
                                    [2]);
```

```
58                          break;
59                  case 'H':
60                          sscanf(line, "%s␣%lf␣%lf␣%lf", str, &H[0], &H[1], &H
                                [2]);
61                          break;
62                  case 'd':
63                          sscanf(line, "%s␣%lf", str, dt);
64                          break;
65                  case 'a':
66                          sscanf(line, "%s␣%lf", str, alpha);
67                          break;
68                  case 'g':
69                          sscanf(line, "%s␣%lf", str, gamma);
70                          break;
71                  case 'l':
72                          sscanf(line, "%s␣%lf", str, loops);
73                          break;
74                  case 'k':
75                          sscanf(line, "%s␣%lf", str, ku);
76                          break;
77                  case 'p':
78                          sscanf(line, "%s␣%d", str, plots);
79                          break;
80                  default:
81                          break;
82                  }
83          }
84
85          /* デバッグ用
86          printf("mx: %lf, my: %lf, mz: %lf\n", m[0], m[1], m[2]);
87          printf("Hx: %lf, Hy: %lf, Hz: %lf\n", H[0], H[1], H[2]);
88          printf("dt: %lf\n", *dt);
89          printf("alpha: %lf\n", *alpha);
90          printf("gamma: %lf\n", *gamma);
91          printf("loops: %lf\n", *loops);
92          */
93          return 0;
94  }
95
96
97  // ----------------------------------------
98  // ルンゲクッタ法
99  // ----------------------------------------
100 int RK4(double* m, double* H, double dt, double alpha, double gamma, double
        ku) {
101         double k1[3], k2[3], k3[3], k4[4];
```

```
102             double m0[3];
103             int i = 0;
104             for (i = 0; i < 3; i++) {
105                     m0[i] = m[i];
106             }
107             Euler(m, H, k1, dt, alpha, gamma, ku);
108             vadd(m, m0, k1, 0.5);
109             Euler(m, H, k2, dt, alpha, gamma, ku);
110             vadd(m, m0, k2, 0.5);
111             Euler(m, H, k3, dt, alpha, gamma, ku);
112             vadd(m, m0, k3, 1.0);
113             Euler(m, H, k4, dt, alpha, gamma, ku);
114             vadd4(m, m0, k1, k2, k3, k4);
115     }
116
117     int Euler(double* m, double* H, double* k, double dt, double alpha, double
            gamma, double ku) {
118             double mx = m[0], my = m[1], mz = m[2];
119             double Hx, Hy, Hz;
120             Heff(m, H, &Hx, &Hy, &Hz, ku);
121             llg(k, mx, my, mz, Hx, Hy, Hz, dt, alpha, gamma);
122             return 0;
123     }
124
125     int llg(double* k, double mx, double my, double mz, double Hx, double Hy,
            double Hz, double dt, double alpha, double gamma) {
126             double c = mx*Hx + my*Hy + mz*Hz;
127             k[0] = dt * (-fabs(gamma)) * (my*Hz - mz*Hy + alpha*(c*mx - Hx)) / (1
                    + alpha*alpha);
128             k[1] = dt * (-fabs(gamma)) * (mz*Hx - mx*Hz + alpha*(c*my - Hy)) / (1
                    + alpha*alpha);
129             k[2] = dt * (-fabs(gamma)) * (mx*Hy - my*Hx + alpha*(c*mz - Hz)) / (1
                    + alpha*alpha);
130             return 0;
131     }
132
133     // ----------------------------------------
134     // H に関する計算
135     // ----------------------------------------
136     int Heff(double* m, double* H, double* Hx, double* Hy, double* Hz, double
            ku) {
137             Hext(H, Hx, Hy, Hz);
138             Hanis(m, H, Hx, Hy, Hz, ku);
139     }
140
141     int Hext(double* H, double* Hx, double* Hy, double* Hz) {
```

```
142             *Hx = H[0];
143             *Hy = H[1];
144             *Hz = H[2];
145     }
146
147     int Hanis(double* m, double* H, double* Hx, double* Hy, double* Hz, double
            ku) {
148             *Hz += 2 * ku * m[3] / sqrt(m[0]*m[0] + m[1]*m[1] + m[2]*m[2]);
149     }
150
151     // ----------------------------------------
152     // vadd
153     // ----------------------------------------
154     int vadd(double* m, double* m0, double* k, double r) {
155             int i = 0;
156             for(i = 0; i < 3; i++) {
157                     m[i] = m0[i] + k[i]*r;
158             }
159
160             double M = sqrt(m[0]*m[0] + m[1]*m[1] + m[2]*m[2]);
161             for(i = 0; i < 3; i++) {
162                     m[i] = m[i]/M;
163             }
164             return 0;
165     }
166
167     int vadd4(double* m, double* m0, double* k1, double* k2, double* k3, double
            * k4) {
168             int i = 0;
169             for (i = 0; i < 3; i++) {
170                     m[i] = m0[i] + (k1[i] + 2*k2[i] + 2*k3[i] + k4[i])/6;
171             }
172
173             double mabs = sqrt(m[0]*m[0] + m[1]*m[1] + m[2]*m[2]);
174             for(i = 0; i < 3; i++) {
175                     m[i] = m[i]/mabs;
176             }
177             return 0;
178     }
179
180
181     // ----------------------------------------
182     // 出力用関数
183     // ----------------------------------------
184     int printout2(double* m, int t) {
185             int i = 0;
```

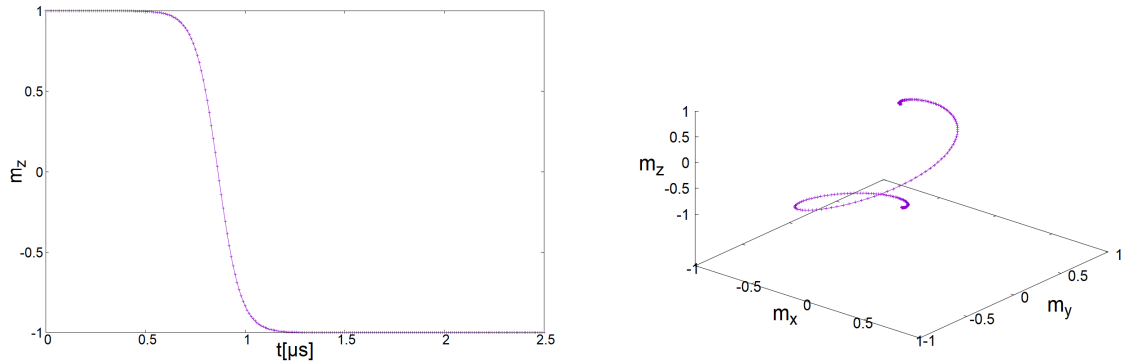
```
186             double ms = (double)t*1.0e-6;
187             printf("%lf␣%.6lf", ms, m[2]);
188             printf("\n");
189             return 0;
190     }
191
192     int printout3(double* m, int t) {
193             int i = 0;
194             for (i = 0; i < 3; i++) {
195                     printf("%.6lf␣", m[i]);
196             }
197             printf("\n");
198             return 0;
199     }
```

このプログラムの使用方法は、まずは同階層に置いた設定用ファイル「init.data」に必要な初期条件を指定、そしてプログラムを実行すると、指定した条件からシミュレーションが実行される。なお、インクルードしている「solver_RK4.h」はプロトタイプ宣言用に用いている。

結果の出力方法は、標準出力へとなっている。これをテキストファイルに保存して、gnuplot で表示することで、結果がグラフで示される。

2.2.3 結果

次に実際に実行した際の結果を以下の図**??**示す。



(a) 結果を 2 次元グラフで表した図　　(b) 結果を 3 次元グラフで表した図

図 1: 小問 1 の結果のグラフ

2.3 小問 2

2.3.1 問題内容

この小問の問題内容は、損失定数を 0.1 及び 0.01 とし、外部磁界 $(0.001, 0, -1)$Oe を加えた場合の原子磁気モーメントの反転シミュレーションを行い、結果を 2 次元および 3 次元のグラフで示す。但し時間刻みは

1ps、磁気回転比は $1.76 \times 10^7 \mathrm{rad}/(\mathrm{Oe} \cdot \mathrm{s})$ なお、計算の都度原子磁気モーメントの再規格化を行うこと。というものとなっている。

2.3.2　結果

使用したプログラムは小問 1 と同様のもののため、記載は省略する。次に実際に実行した際の結果を以下の図 2 および図**??**に示す。

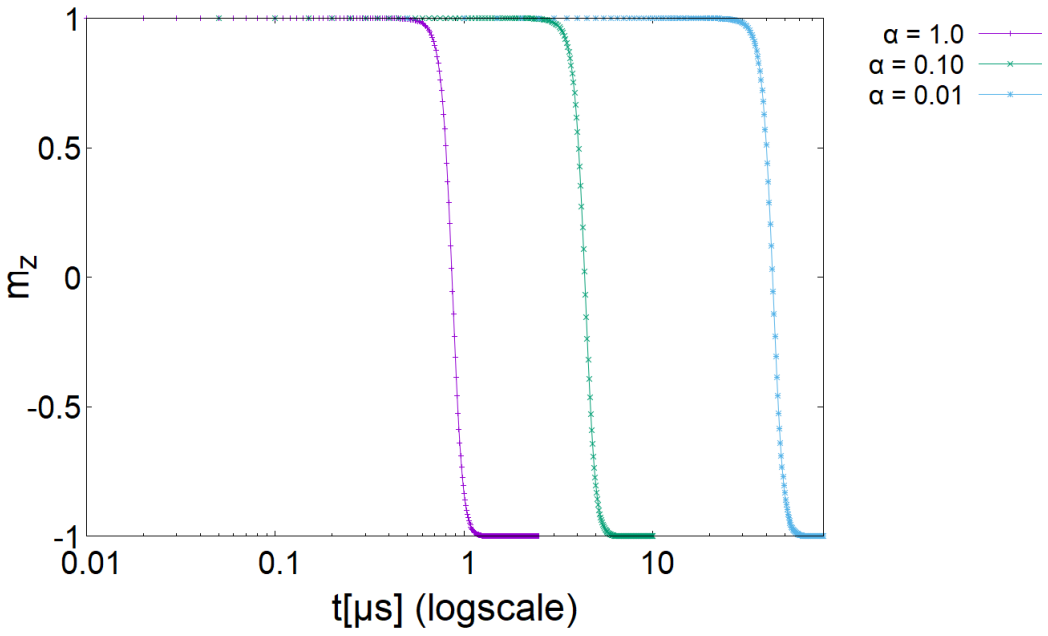
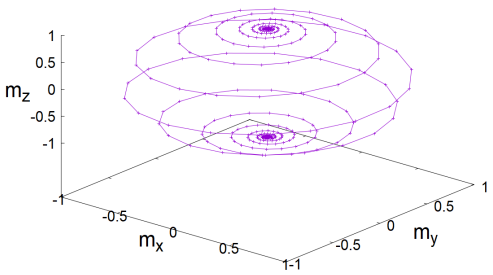
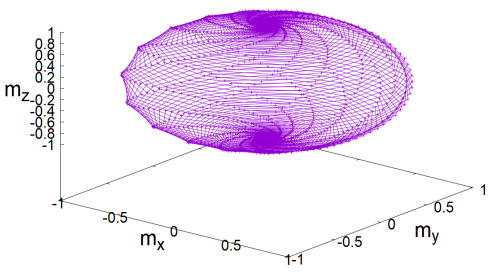


図 2: 各反転シミュレーションの結果を 2 次元グラフで表した図



(a) $\alpha = 0.1$ の時の結果を 3 次元グラフで表した図



(b) $\alpha = 0.01$ の時の結果を 3 次元グラフで表した図

図 3: 小問 2 の結果の 3 次元グラフ

2.4　小問 3

2.4.1　問題内容

この小問の問題内容は、損失定数を 0.001 から 100 まで変化させて、反転時間 t_{sw} の変化を調べる事となっている。

2.4.2　結果

使用したプログラムは小問 1 のものと大きく変わらず、出力する内容を t_{sw} に絞ったプログラムとなったため、記載を省略する。

今回は損失定数が 0.001, 0.003, 0.1, 0.3, 1, 3, 10, 30, 100 の時の t_{sw} を調べ、それをグラフにまとめた。以下の図 4 がそのグラフだ。

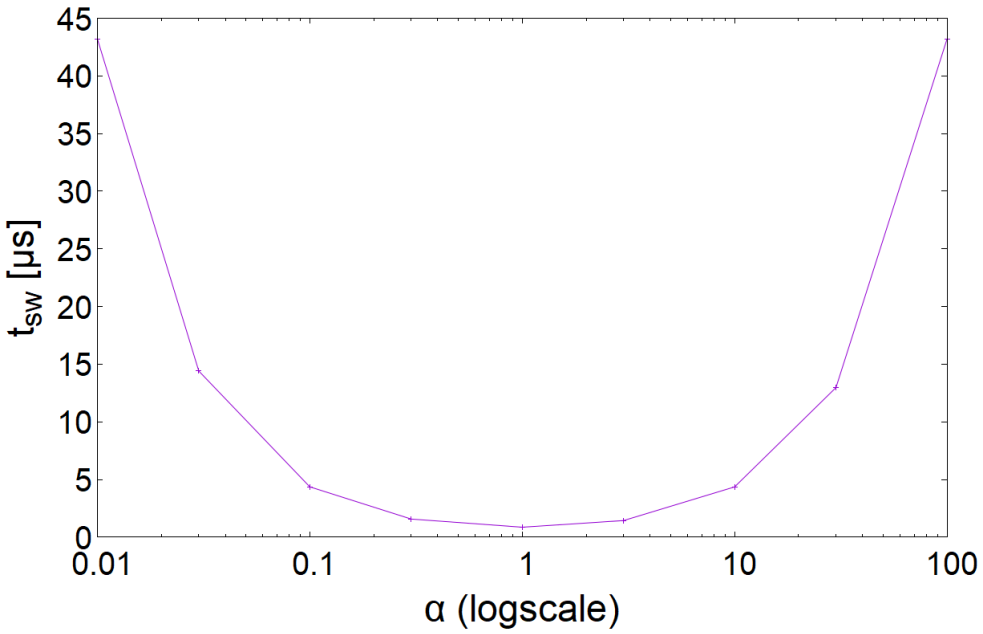


図 4: 損失定数と反転時間の関係を表すグラフ

ここから、損失定数が 1 の時が最も早く反転し、1 から離れれば離れるほど反転時間が長くなる。

3　問題 2

3.1　原理

3.1.1　一軸磁気異方性

一軸磁気異方性とは、磁化がある一つの軸方向に向いている時にエネルギーが減少する性質を指す。このある一つの軸のことを磁化容易軸と呼ぶ。z 軸が磁化容易軸である場合の一軸異方性エネルギーは以下の 6 式で表される。

$$\varepsilon^K = K_u(1 - m_z^2) \tag{6}$$

ここで K_u は異方性定数であり、単位は erg/cm^3 となる。

このエネルギーを変分することで、エネルギーから磁界への換算が行える。

$$\vec{H} = -\frac{\delta\varepsilon}{\delta\vec{M}} \tag{7}$$

一軸磁気異方性エネルギーの場合、この変換は 8 式で表わされる。

$$\vec{H}^K = -\frac{\delta\varepsilon^K}{\delta\vec{M}} = \begin{pmatrix} 0 \\ 0 \\ \frac{2K_u}{m} m_z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ H_k m_z \end{pmatrix} \tag{8}$$

ここで H_k は異方性磁界と呼ばれる。

3.2 小問 1

3.2.1 問題内容

この小問の問題内容は、磁化容易軸が z であり、異方性磁界が 10kOe の一軸磁気異方性磁界を考える。次に、原子磁気モーメントを x-z 平面内で z 軸から 45 度傾いた方向 $(0, 1/\sqrt{2}, 1/\sqrt{2})$ に向け、その後の磁気モーメントの運動を調べる事となっている。また、損失定数は 0.01、磁気回転比は $1.76 \times 10^7 \mathrm{rad/(Oe \cdot s)}$、時間刻みは 1ps で計算を行う。

3.2.2 結果

計算を行うプログラムは問題 1 と変わらず、初期条件のみを変更したため省略する。

以下に問題内容に沿った条件でシミュレーションした結果のグラフを、図 5 として示す。

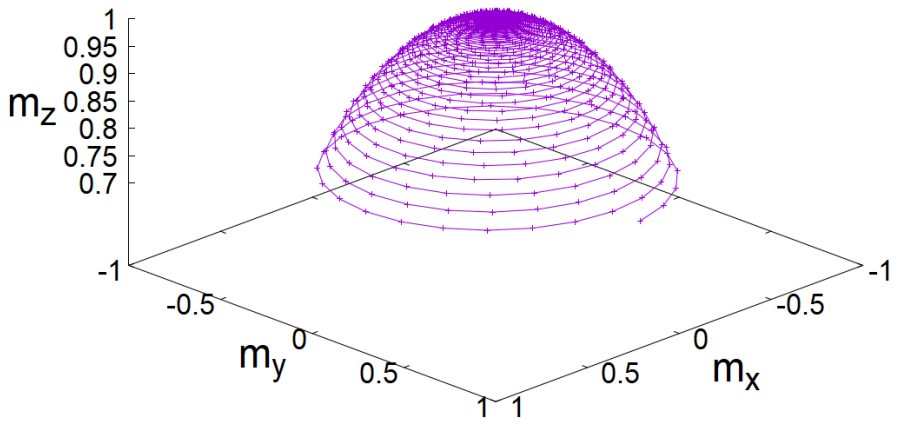


図 5: 小問 1 の結果のグラフ

このグラフから、初期地点から歳差運動をしつつ一軸磁気異方性磁界の方向へと近づいているのが分かる。

3.3　小問 2

3.3.1　問題内容

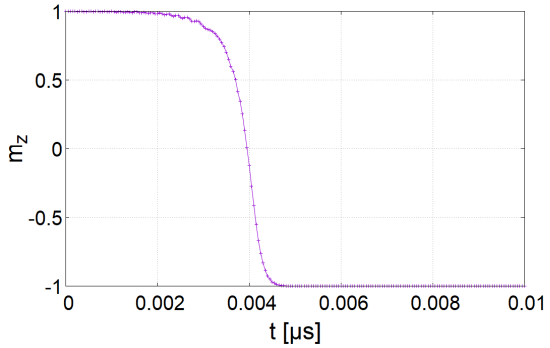
この小問の問題内容は、原子磁気モーメントを z 軸方向 $(0,0,1)$ へ向け、z 軸から 0.01 度傾いた 15kOe の外部磁界 $(15\mathrm{k}\times\sin(0.01),0,-15\mathrm{k}\times\cos(0.01))$ を加え、反転の様子を調べる。そして、時間刻みを変化させて反転の様子を再度調べる事となっている。

また、上で触れなかった初期条件については小問 1 を踏襲することとする。

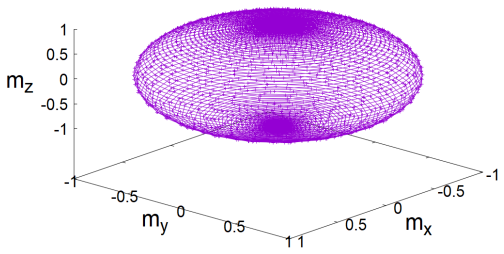
3.3.2　結果

計算を行うプログラムは小問 1 と変わらず、初期条件のみを変更したため省略する。

まず、以下に時間刻みを変化させない状態 $dt = 1\mathrm{ps}$ でシミュレーションした結果を、2 次元と 3 次元のグラフで図**??**として表示する。



(a) 結果を 2 次元グラフで表した図



(b) 結果を 3 次元グラフで表した図

図 6: $dt = 1\mathrm{ps}$ の結果のグラフ

このグラフから、原子磁気モーメントが一軸磁気異方性磁界に逆らい、外部磁界の方に寄って行っているのが分かる。これは一軸磁気異方性磁界よりも外部磁界のほうが大きいによりこのような運動をしている。

次に、時間刻みを変化させて反転の様子を調べた結果について。時間刻みは、まず 1ps から 1ps ずつ増加させて計算を行っていった。このとき 8ps から原子磁気モーメントは反転をしなくなった。これは更に時間刻みを変化させても反転することはなかった。

そして、時間刻みを 1ps から順に、1/10 の値を計算していった。しかし、0.1fs まで計算をしたが、変化はなかった。

以下に 1, 5, 8, 0.1ps, 0.1fs の時の反転の様子をまとめたグラフを、図 7 として示す。

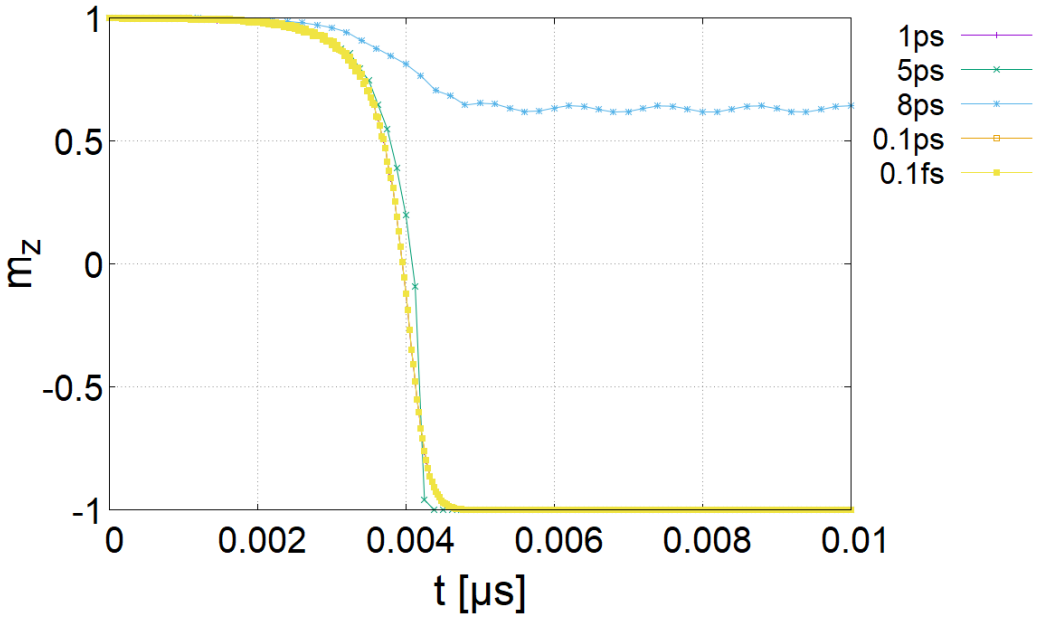


図 7: 時間刻みを変化させた時の反転の様子

この結果から、今回の初期条件においては時間刻みを 8ps 以上の値に設定すると反転しなくなる。つまり、計算が破綻することがわかった。また、時間刻みを小さくしても計算が破綻することはなかった。

3.4 小問 3

3.4.1 問題内容

この小問の問題内容は、外部磁界を変化させて計算を行い、外部磁界と反転時間の関係を調べる。また、原子磁気モーメントが反転する最小の磁界を求める事となっている。

また、上で触れなかった初期条件については小問 2 を踏襲することとする。

3.4.2 結果

外部磁界を、1.0kOe から 0.1kOe ずつ増加させて反転時間を調べていった。このとき、原子磁気モーメントが反転する最小の磁界は 9.2kOe となった。

以下に横軸が外部磁界で、縦軸が反転時間のグラフを、図 8 として示す。

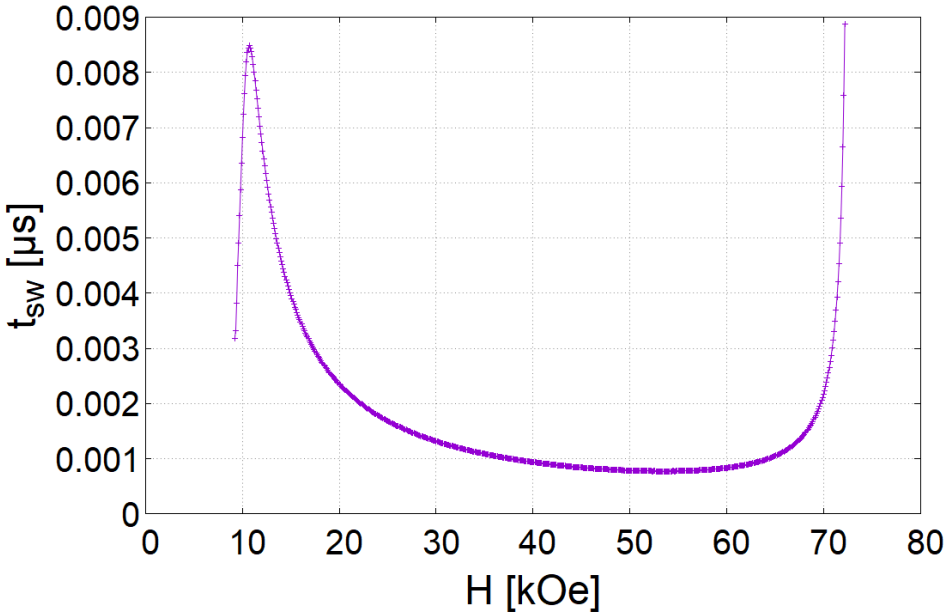


図 8: 外部磁界を変化させた時の反転時間の様子

また、このグラフから原子磁気モーメント最も早く反転する磁界は 53.4kOe だということも分かる。

4 参考文献

- 配布された LLG のテキスト