平成 24 年度

卒業研究報告書

研究題目

マイケルソン干渉計を用いた波長計の製作

富山高等専門学校

電子制御工学科　堺洋貴　村田義光

指導教官　由井四海

平成 25 年 3 月 1 日　提出

目次

1. はじめに

レーザー光を使用する場合に、そのレーザー光の波長を知っておくことは非常に重要である。波長を知るために波長計が必要となるが、とても高価なものである。そこで干渉計を用いて波長計の製作を行う。干渉計にはフィゾー干渉計やマッハツェンダー干渉計などがあるが、本研究では検出感度が良く、構成が簡単なマイケルソン干渉計を用いた波長計の製作を目的とした。

2.　研究概要

2.1.　光の干渉について

干渉とは、二つ以上の光波が光路差を経て一点で重なる時、その点での波の振幅がそれぞれの波の振幅の和で表されることである。光路差が半波長変わるごとに光は明暗を繰り返す。光路差が奇数倍のとき光は弱め合い、偶数倍のとき光は強め合うという性質がある。

図 1 に光路差が半波長の奇数倍のときに干渉した図を示す。

図 2 に光路差が半波長の偶数倍のときに干渉した図を示す。

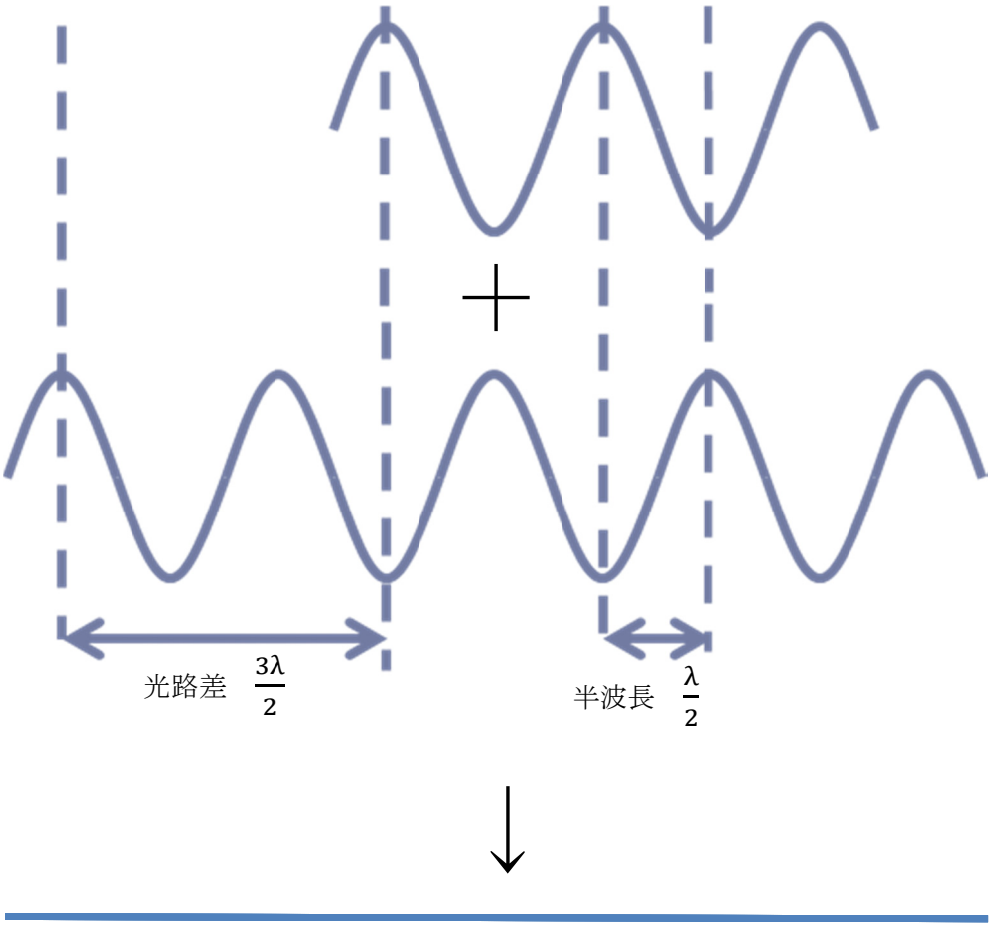

図 1　光路差が半波長の奇数倍

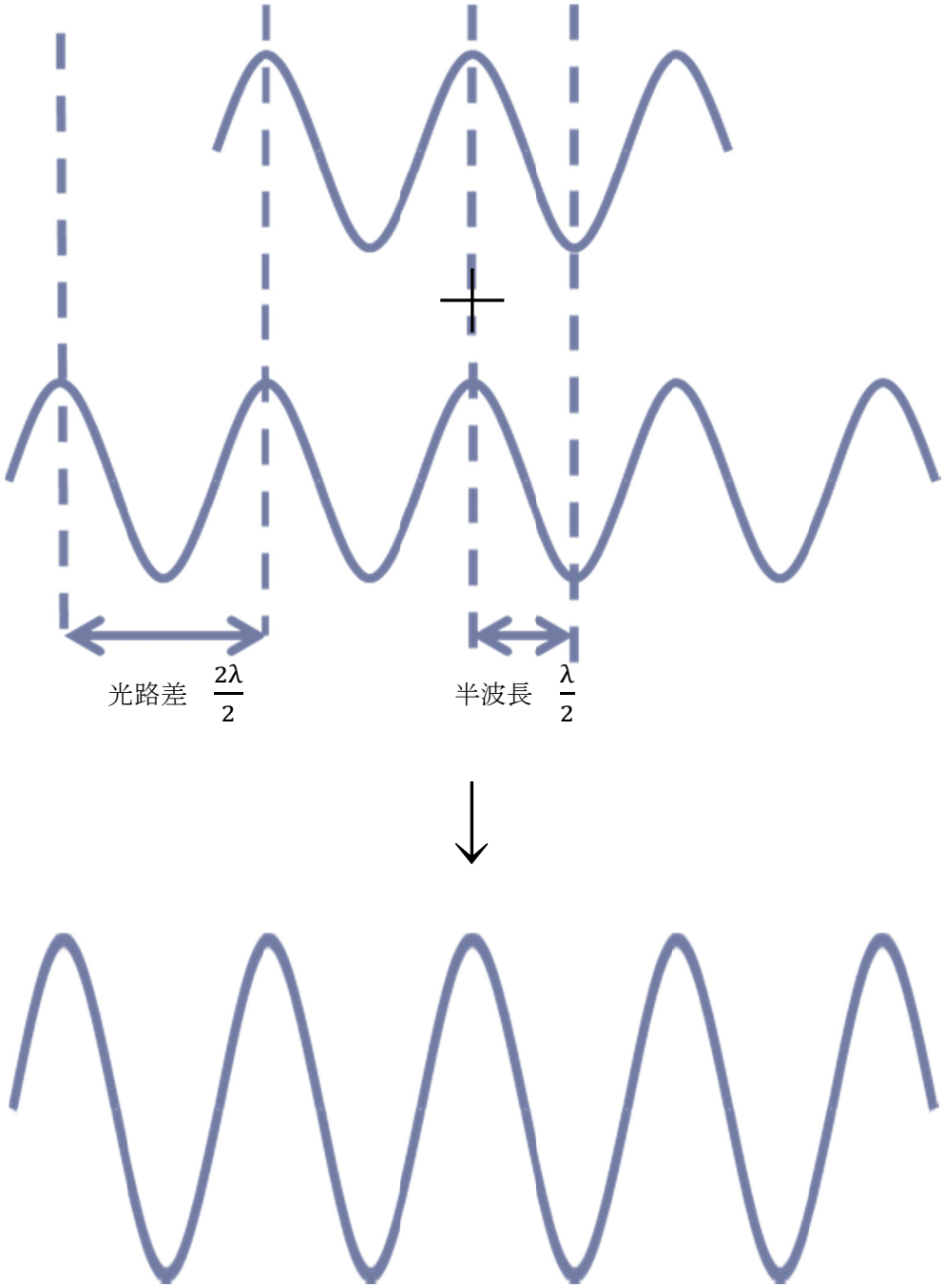

図 2　光路差が半波長の偶数倍

2.2. 干渉計の種類

以下に干渉計の種類とそれぞれの簡単な特徴を説明する。

・マイケルソン干渉計

検出感度が良い、構成が簡単

・フィゾー干渉計

振動に強い、コンパクトに構成できる

・マッハツェンダー干渉計

大型測定物用の干渉計として利用できる

・ジャマン干渉計

気体の屈折率を測定するときに用いる

・ファブリー・ペロー干渉計

バンドパスフィルタなどに用いる

2.3.　マイケルソン干渉計の原理

マイケルソン干渉計の構成を図 3 に示す。レーザー光を分岐比が 50:50 のビームスプリッターで２つに分け、それぞれをミラーで反射させビームスプリッターで重ね合わせることで、フォトダイオード(Thorlabs 社、PDA100A)上で干渉縞が生じる。ミラーを移動させ、光路差を変化させることで光が強め合ったり弱め合ったりする。光が強め合う回数もしくは弱め合う回数を n、光路差の変化を x とすると、波長 λ は以下の式(1)で求めることができる。本研究では、強め合う回数を n とした。

$$\lambda = \frac{x}{n} \quad \cdots (1)$$

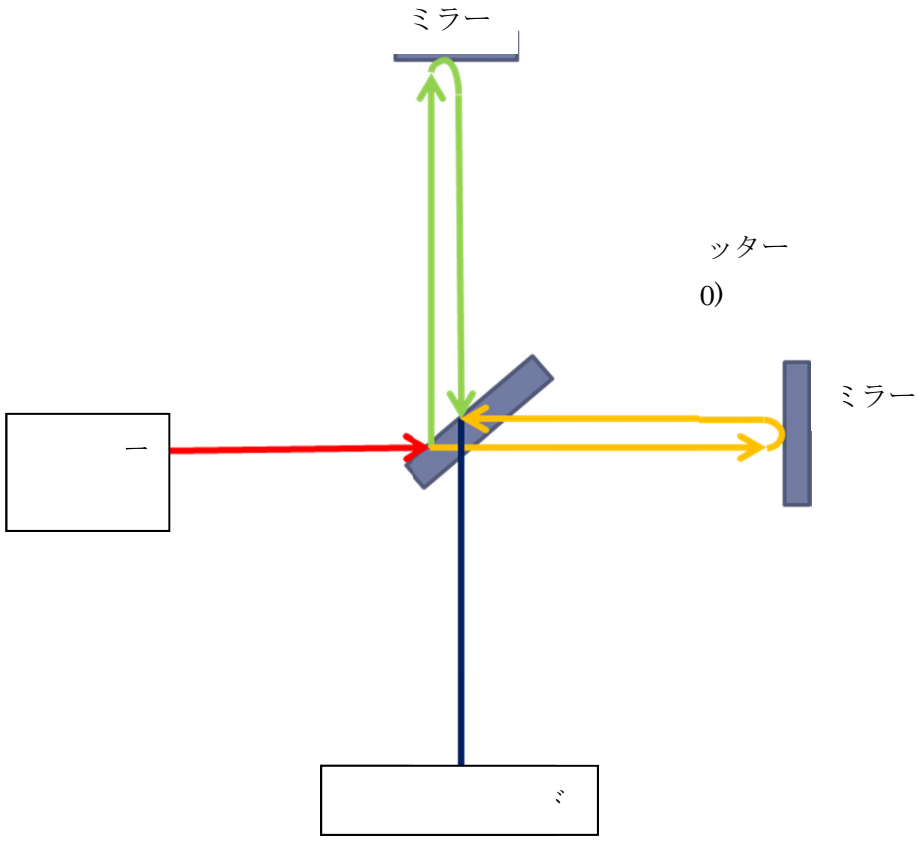

図 3　マイケルソン干渉計

3.　実験内容

3.1.　実験方法

干渉計を用いてレーザー光の波長を測定する。レーザー光を干渉させるためマイケルソン干渉計を作製した。片方のミラーを移動させることで光路差を変え、光を強め合ったり弱め合ったりさせる。そのときの強め合う回数と光路差の変化を(1)式に代入し、波長を求める。光が強め合う点は出力波形のピークであるから、ピークの数 n を求めるプログラムを作製した。プログラムのフローチャートを図 4 に示す。

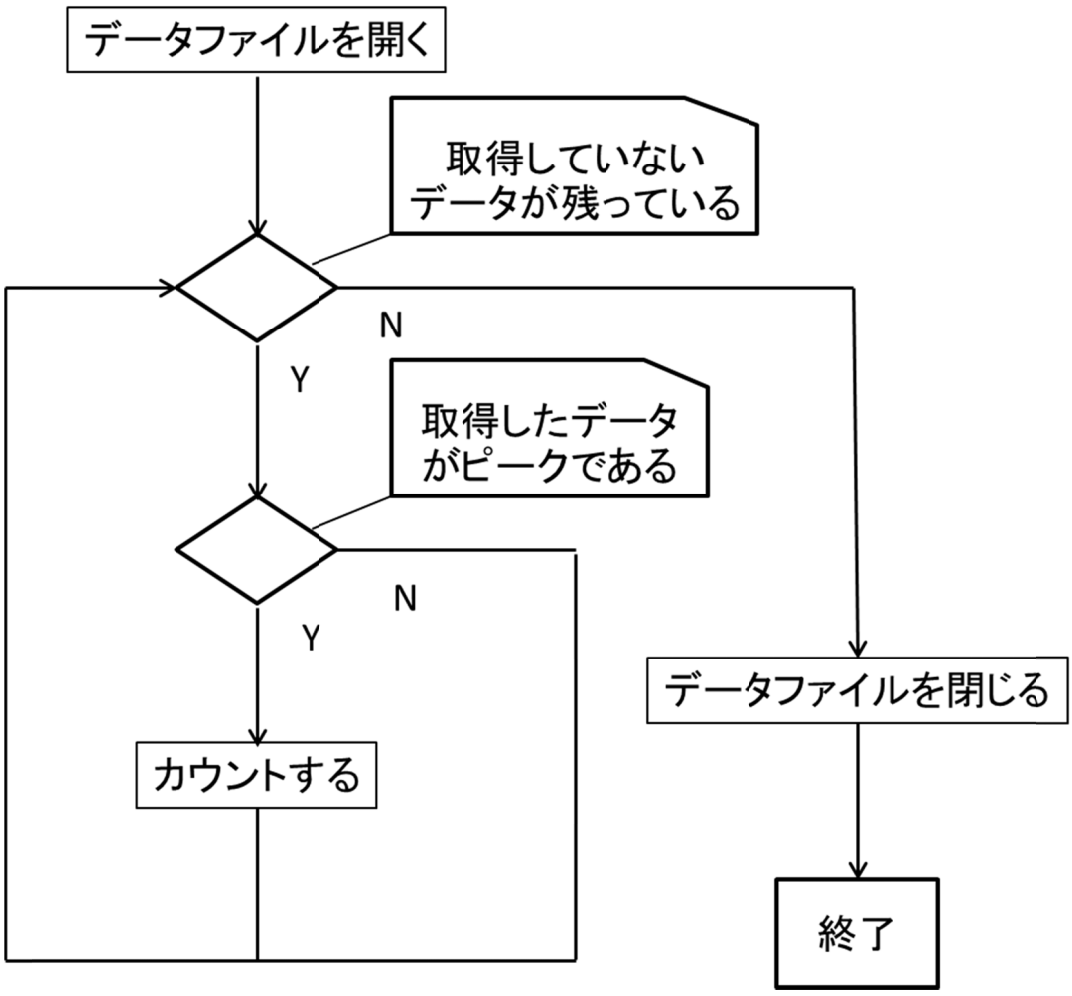

図 4　プログラムのフローチャート

3.2.　He-Ne レーザー光の波長の測定

3.2.1.　実験装置

He-Ne レーザー光を分岐比 50:50 のビームスプリッター(Thorlabs 社)で２つに分け、分けられた光をそれぞれミラーで反射させ、一点に重ねることでディテクター上に干渉縞ができる。ミラーの片方を 1mm 移動させて光路長を変え、光を干渉させる。

実験系を図 5 に示す。

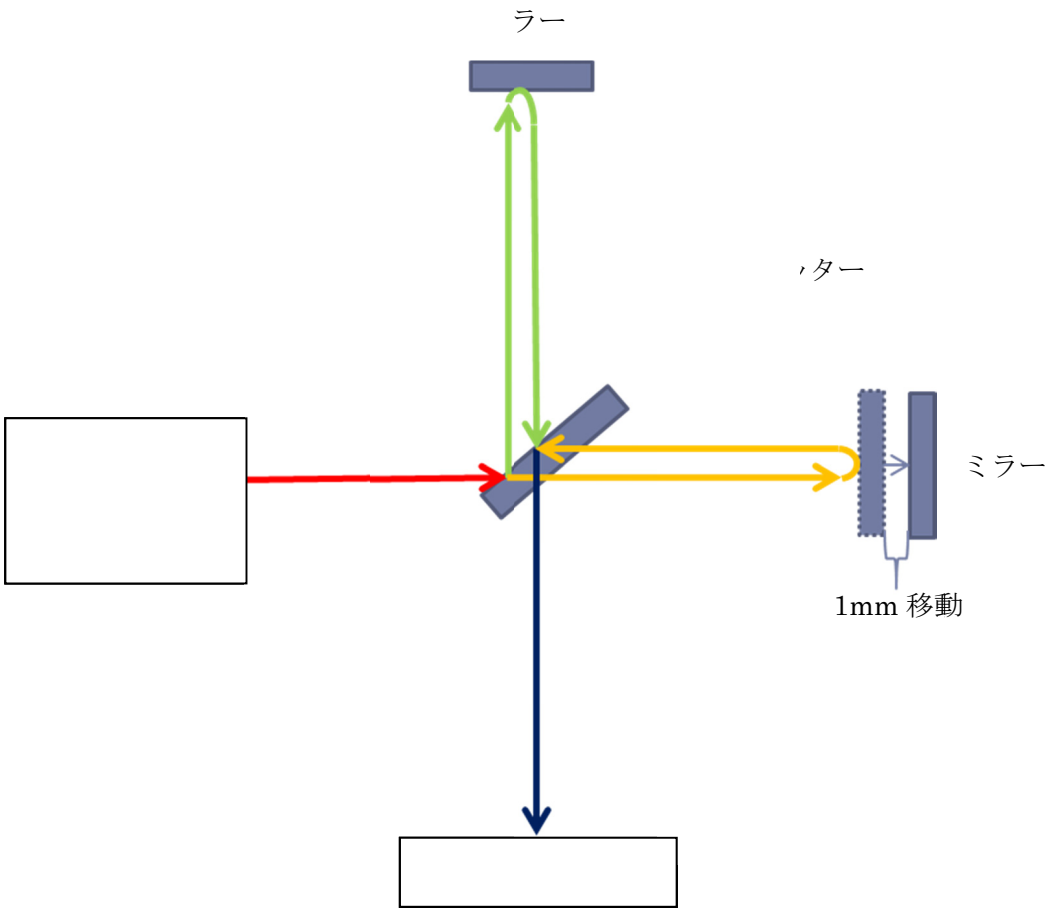

図 5　１波長の場合の実験装置

3.2.2.　ミラーの移動方法

ミラーをローベアリングリニアステージ(シグマ光機、SGSP15-10)に固定して 1ｍｍの金属板を図のように差し込み、抜くことで移動させた。同時にこの時、距離センサ(オムロン、ZX-LDA11-N 2M)を使用して、移動距離を測定した。

図 6 にミラーの移動方法を示す。

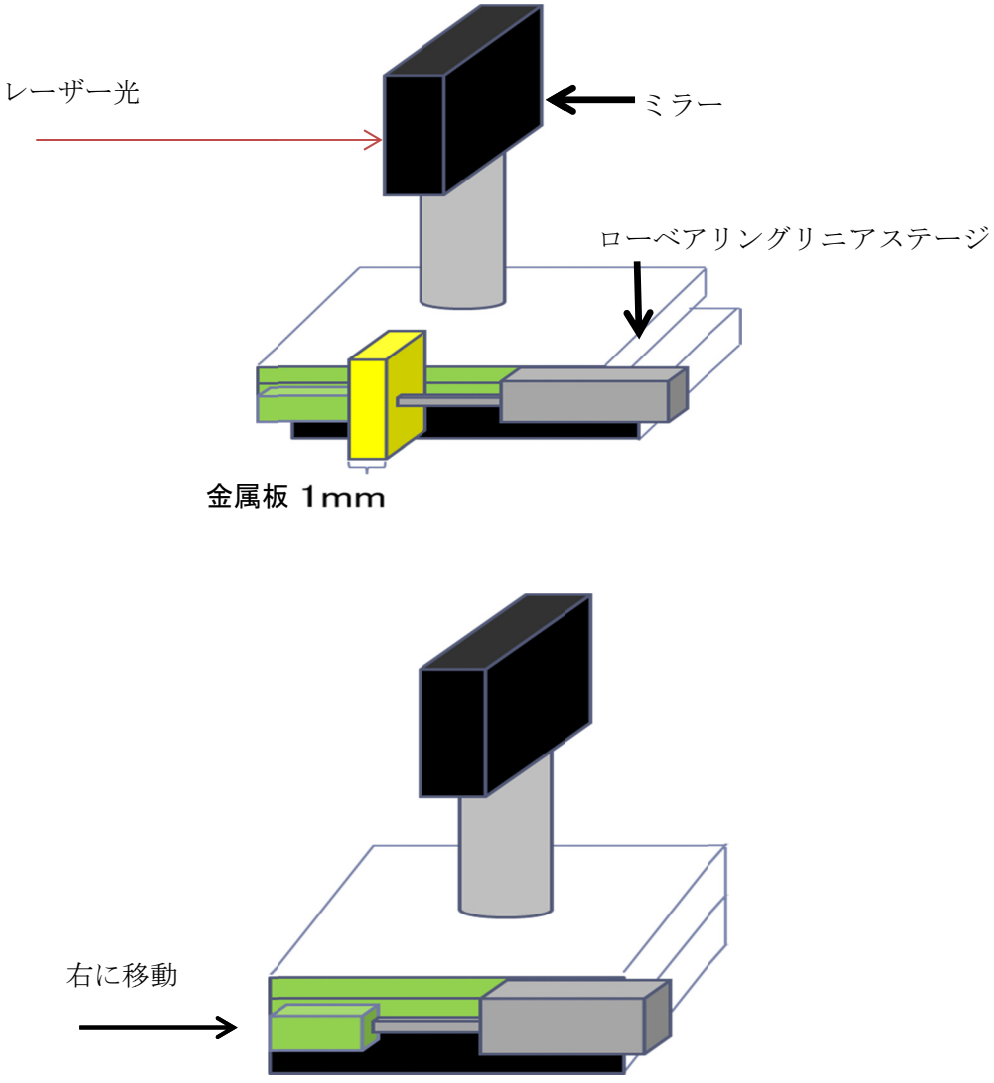

図 6　ミラーの移動方法

3.2.3. 実験結果

図 7 に He-Ne レーザー光を干渉させたときの測定結果を示す。

黒い波形は干渉波形を示し、灰色の波形は距離センサの出力を示している。距離センサの出力電圧が変化している間干渉している。

図 8 に He-Ne レーザー光を干渉させたときの測定結果の拡大図を示す。

光が強め合ったとき、出力波形はピークの値をとる。

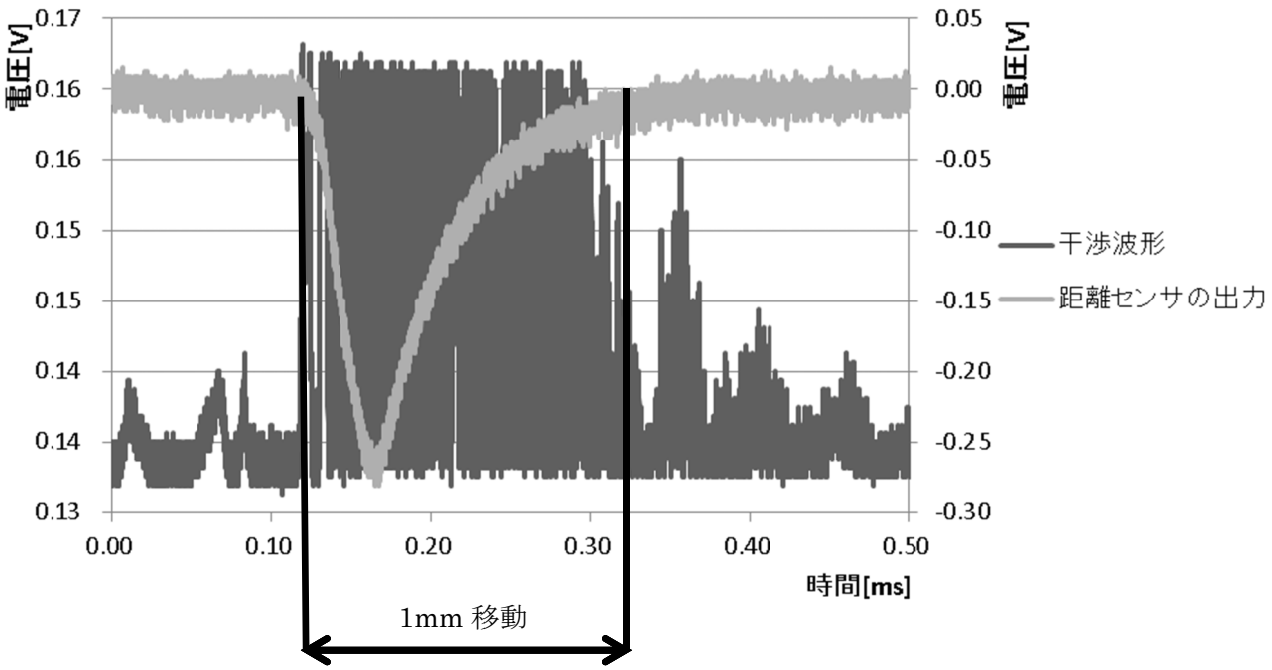

図 7　He-Ne レーザー光を干渉させたときの測定結果

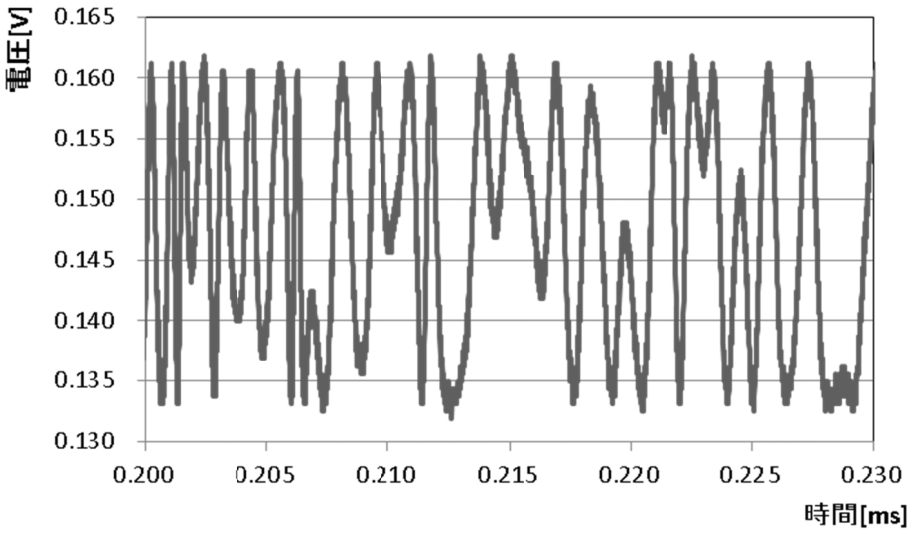

図 8　He-Ne レーザー光を干渉させたときの測定結果の拡大図

図 7 の干渉波形において、1mm 移動したときに光が強め合った回数をプログラムで数えると、1664 回となった。この結果を(1)式に代入すると、波長は 601.0nm となる。He-Ne レーザー光の波長は 632.8nm であるため、測定結果とは約 32nm 短いことがわかる。この原因としては、移動距離が 1mm ではなかったことだと考えられる。そこで、波長を正確に測定するために二波長のレーザー光を使用できるマイケルソン干渉計を作製する。レーザー光の種類は、波長が分かっている He-Ne レーザー光と、波長を測りたい半導体レーザー光にする。He-Ne レーザー光を a、半導体レーザー光を b とすると、b の波長λ_bは以下の式で求めることができる。 He-Ne レーザー光の波長をλ_a、He-Ne レーザー光が強め合った回数をn_a、半導体レーザー光が強め合った回数をn_b、移動距離を x とする。

(1)より　$x = \lambda_a * n_a$・・・(2)

$$\lambda_{\mathrm{b}} = \frac{x}{n_b} \cdot \cdot \cdot (3)$$

(2)を(3)に代入　$\lambda_{\mathrm{b}} = \frac{\lambda_a * n_a}{n_b} \cdot \cdot \cdot (4)$

この式を用いると、移動距離に関わらず波長を求めることができる。

3.3.　He-Ne レーザー光と半導体レーザー光の実験

3.3.1.　実験装置

二つの直角プリズムを左右に移動させることで光路差を変える。

実験装置を図 9 に示す。

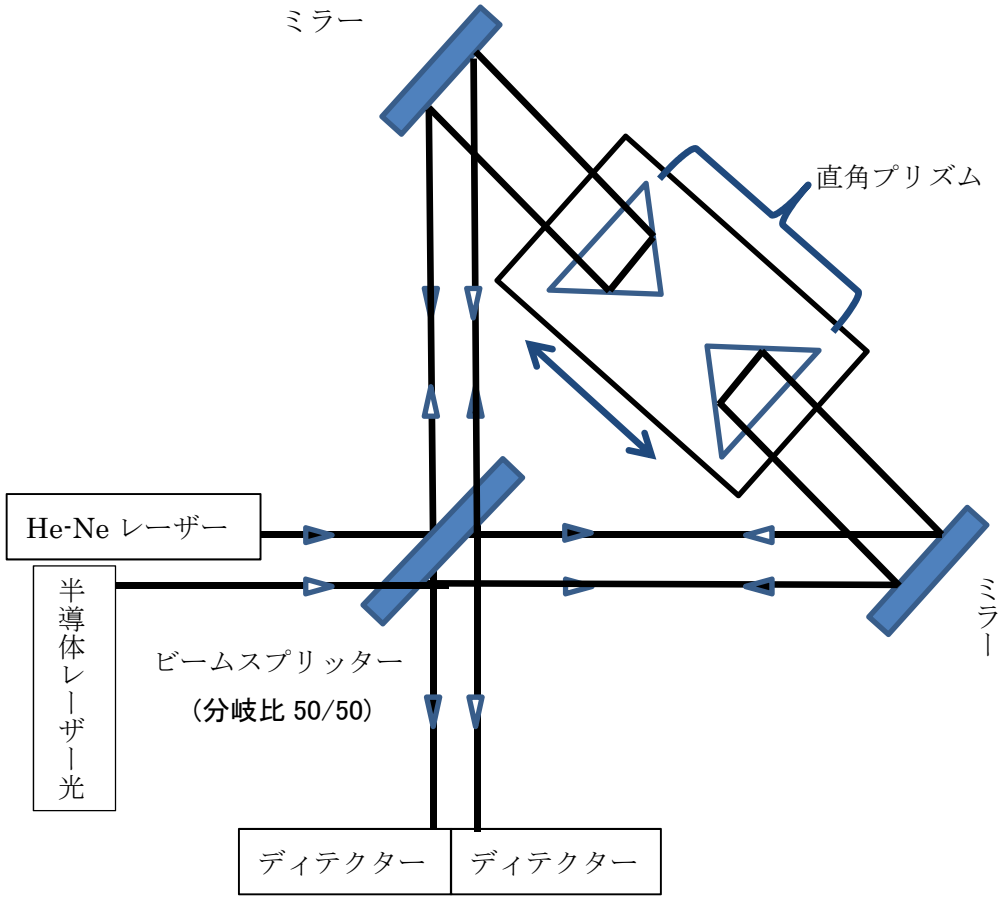

図 9　マイケルソン干渉計による波長計

3.3.2.　直角プリズムの移動方法

二つの直角プリズムをリニアガイドに乗せ、ボイスコイルモータ(Akribis、AVM20-10)で左右に移動させた

図 10 に直角プリズムの移動装置を示す。

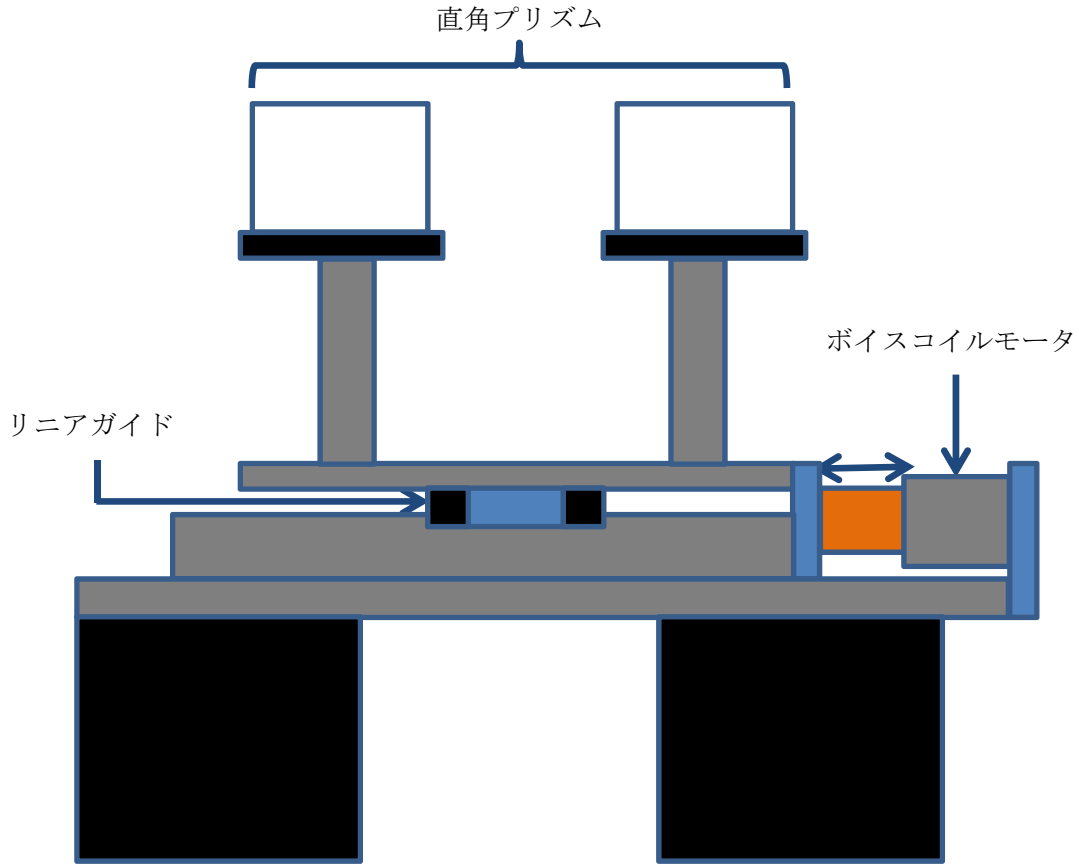

図 10　直角プリズムの移動装置

ボイスコイルモータとは、永久磁石の磁界中のコイルに電流を流すと、フレミングの左手の法則により磁界と電流の双方に垂直方向に力が発生し、直線運動をするモータのことである。

3.3.3. ボイスコイルモータへの電圧の印加方法

ボイスコイルモータに直流電源から電圧を印加すると、モータが一方向にしか動けず、移動距離が短いのでレーザー光の波長を正確に求めるために必要な干渉波形が測定できない。移動距離を確保するため、ボイスコイルモータを左右に動かす必要がある。そのためには、順電圧と逆電圧を交互に印加する必要がある。そこで、波形発振器を用いてボイスコイルモータに電圧を印加することにした。しかし、波形発振器を用いると実験装置が大型化してしまうという欠点がある。また、順電圧と逆電圧を交互にかける場合、どちらかの電圧が少しでも大きいと台が少しずつ移動し、端にぶつかって止まってしまう。その 2 つの課題をを解決するため、マイコンボード(Arduino Uno)を用いてプログラムでボイスコイルモータに印加する電圧を制御する。

マイコンボードに入力したプログラムを以下に示す。

```
int ledPin1 = 13;                    // LED とデジタルピン 13 を接続する
void setup()
{
        pinMode(ledPin1, OUTPUT);        //出力のデジタル端子を設定
}
void loop()                          // 繰り返し
{
        digitalWrite(ledPin1, HIGH);    // LED を on にする
        delay(30);                       // 30ms 待つ
        digitalWrite(ledPin1, LOW);      //LED を off にする
        delay(120);                      // 120ms 待つ
}
```

digitalWrite 関数を使うと、ピンの状態を HIGH か LOW に選択することが出来る。

setup 関数内で pinMode 関数を使って LED が備え付けられたピンを出力ピンに設定する。出力モードにすると、プログラム側からピンの状態を選択することが可能になる。Arduino Duemilanove 備え付けの LED はソース制御型になっており、13 番ピンを HIGH にすれば電流が流れる。

指定のピンの HIGH/LOW を設定するには digitalWrite 関数を使用する。digitalWrite 関数は引数にピン番号と HIGH/LOW をとる。電流を流すには、第 2 引数を HIGH にする。逆に流さないは第 2 引数を LOW にする。これを交互に繰り返すことで、ボイスコイルモータを動かしたり止めたりできる。

ソースコード上で digitalWrite 関数を使い、HIGH と LOW を繰り返す。間に挟まっている delay 関数は指定の時間だけ処理を休む関数で、引数はミリ秒指定になっている。今回は 30ms 電流を流し、120ms 電流を流さないプログラムにした。つまり loop 関数の処理は『30ms 電流を流す→120ms 電流を流さない、を繰り返す』処理になる。

しかし、この方法では台が一方向しか動かない。そこで、OFF 時間に移動台をバネで引っ張ることで、左右に動かすという方法を用いた。そうすることで、装置を小型化し、台を左右に動かすことができた。

3.3.4. 移動台の振動の低減

図 11 にモータ駆動用回路を示す。

移動台をボイスコイルモータで移動させた場合、オンとオフを切り替えてバネで引っ張るとき、台が振動してしまう。台が振動すると光路が変化し、干渉が起きなくなってしまうため、波長を求めることができない。

そこで、図 11 に示すように回路に電解コンデンサを並列に接続することで台の振動を抑えようと試みた。接続するコンデンサを 10 μ F と 470 μ F と 1000 μ F で実験した結果、1000 μ F を使用した時が最も振動を抑えることができたので 1000 μ F のコンデンサを使用することにした。

図 12 に移動台の音の変化の結果を示す。

振動したときに発生する音が、コンデンサを接続すると小さくなったので振動が小さくなったと考えた。

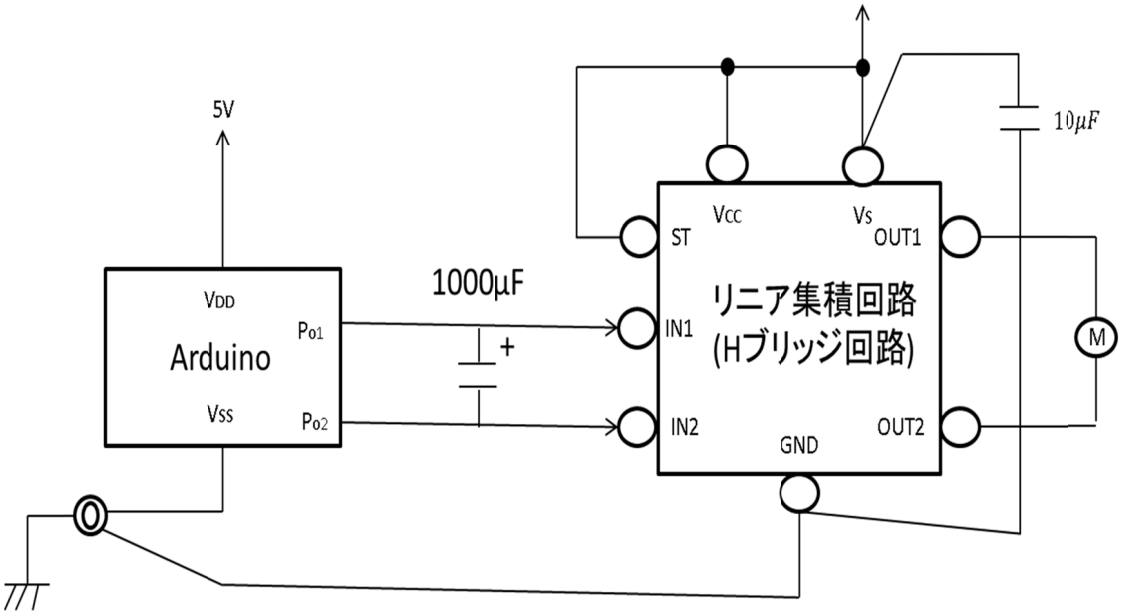

図 11　モータ駆動用回路

リニア集積回路(TOSHIBA、TA8429H/HQ)

コンデンサなし　　コンデンサあり

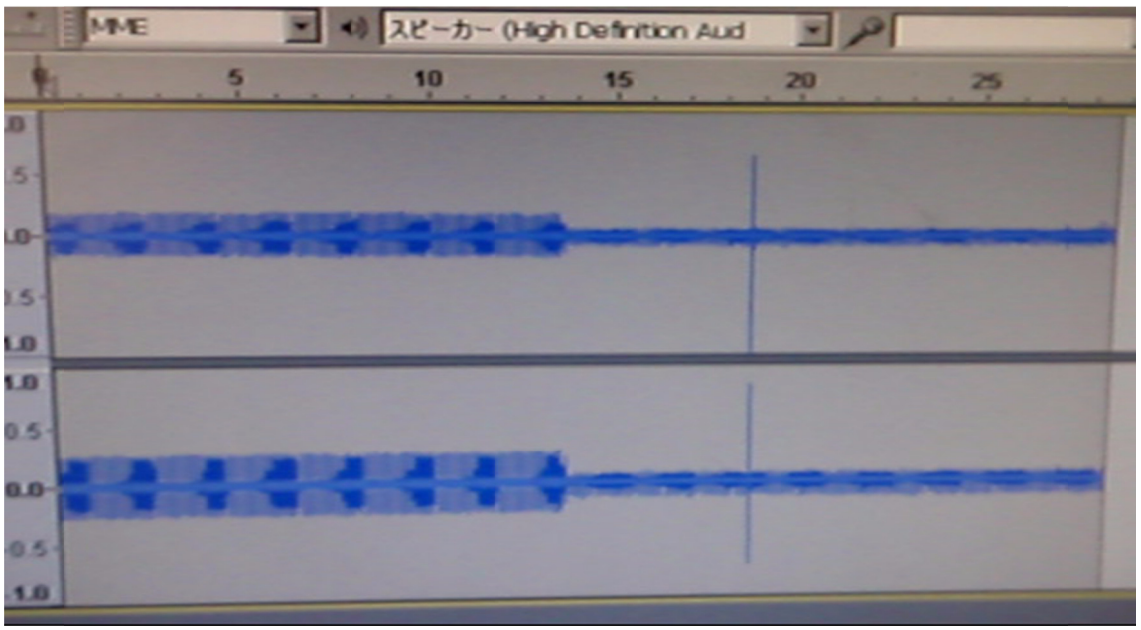

図 12　移動台の音の変化の結果

3.3.5. He-Ne レーザー光での実験

ボイスコイルモータで移動台を動かした場合でも光を干渉させることができるか確認するため、最初に He-Ne レーザー光のみで実験を行った。

図 13 に He-Ne レーザー光を干渉させたときの測定結果を示す。

図 14 に He-Ne レーザー光を干渉させたときの測定結果の拡大図を示す。

この結果を見ると、電圧が時間変化していることから光の強弱が生じていることが分かる。このことから、ボイスコイルモータで光路差を変えても光が干渉することが確認できる。

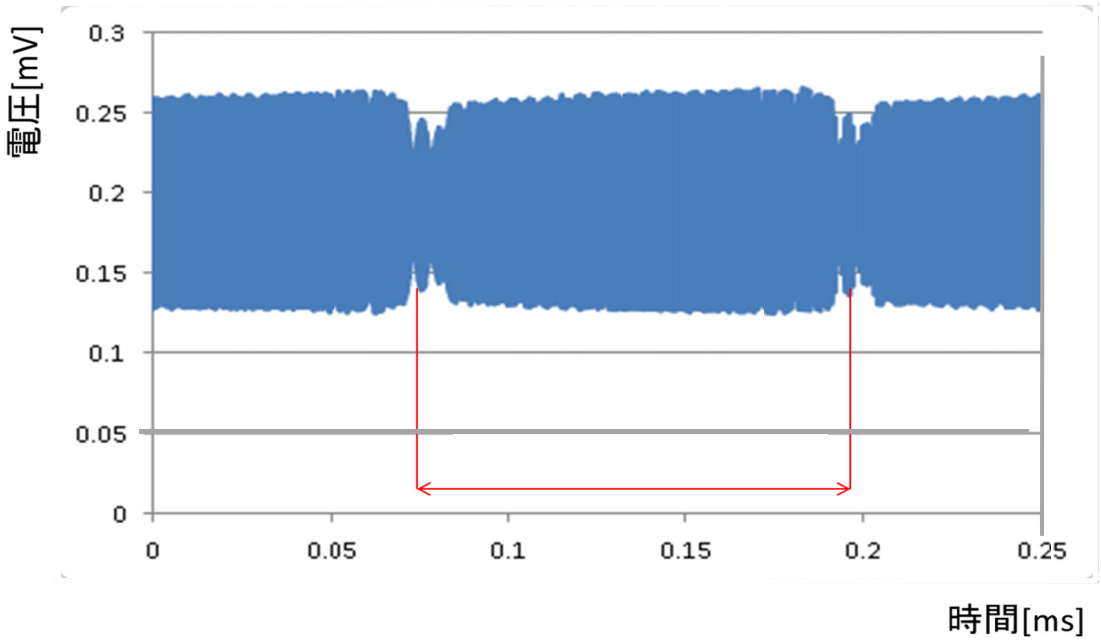

図 13　He-Ne レーザー光を干渉させたときの測定結果

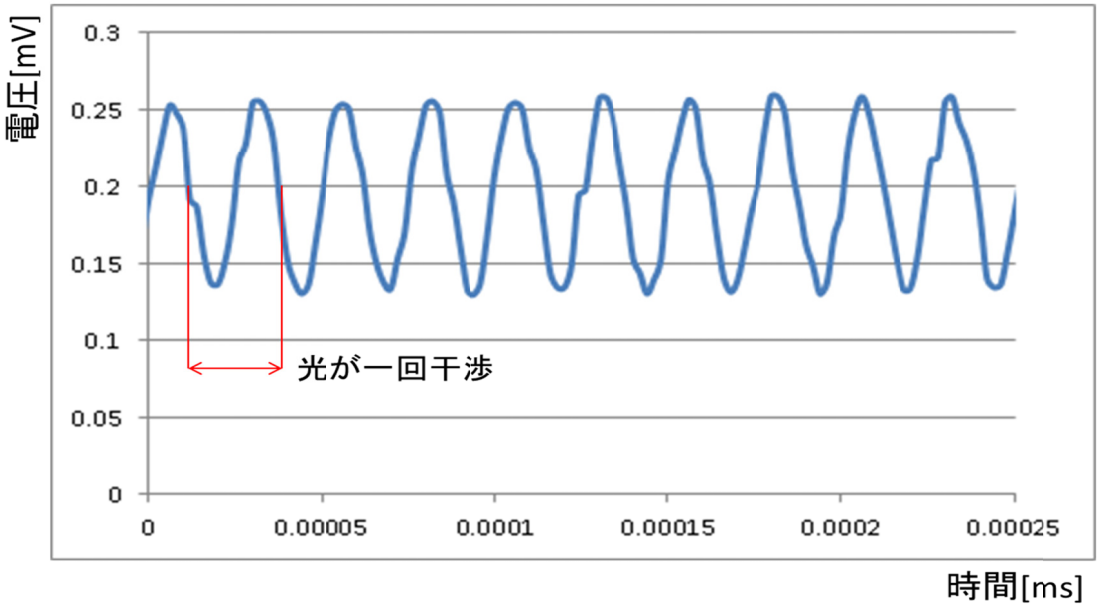

図 14　He-Ne レーザー光を干渉させたときの測定結果の拡大図

3.3.6. He-Ne レーザー光と半導体レーザー光での実験

3.3.6.1. He-Ne レーザー光と 634.0nm の半導体レーザー光での実験

ボイスコイルモータで台を移動させて光が干渉することが確認できたので、波長が既知の He-Ne レーザー光と波長が不明な半導体レーザー光を使用し、半導体レーザー光の波長の導出を試みた。

図 15 に He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果を示す。

黒い波形は He-Ne レーザー光の波形を示し、灰色の波形は半導体レーザー光の波形を示している。波が大きく変化していることから、それぞれ干渉していることがわかる。

図 16 に He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果の拡大図を示す。

拡大図を見ると、He-Ne レーザー光は半導体レーザー光に比べて干渉波形の周期が短いことから、強め合った回数が多く、波長が短いことがわかる。

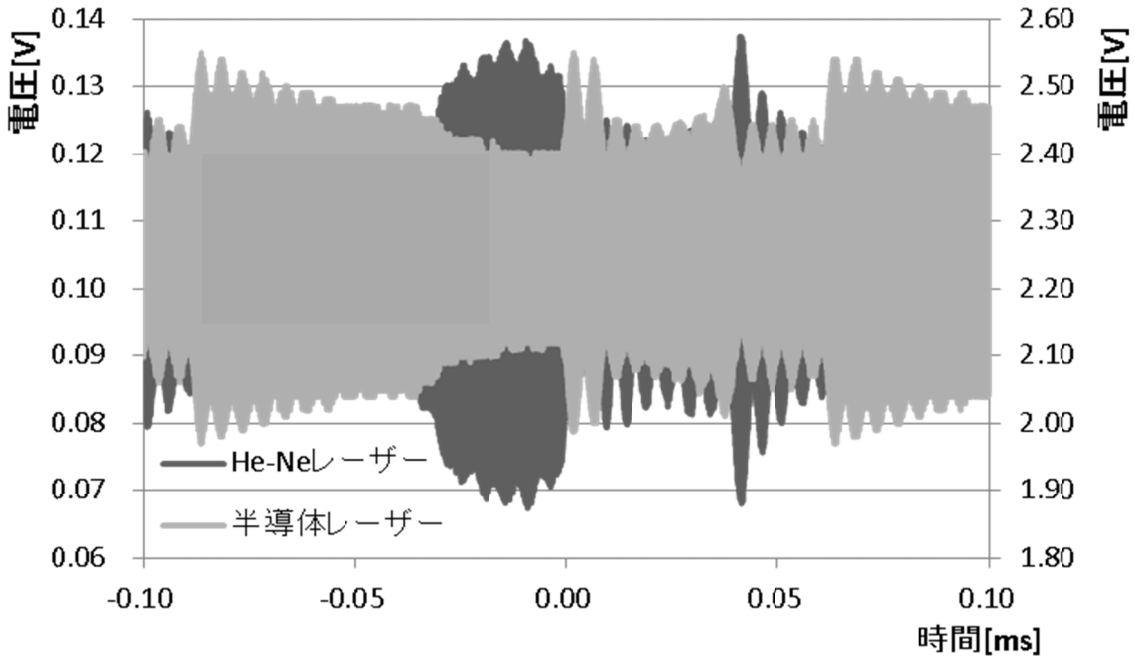

図 15　He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果

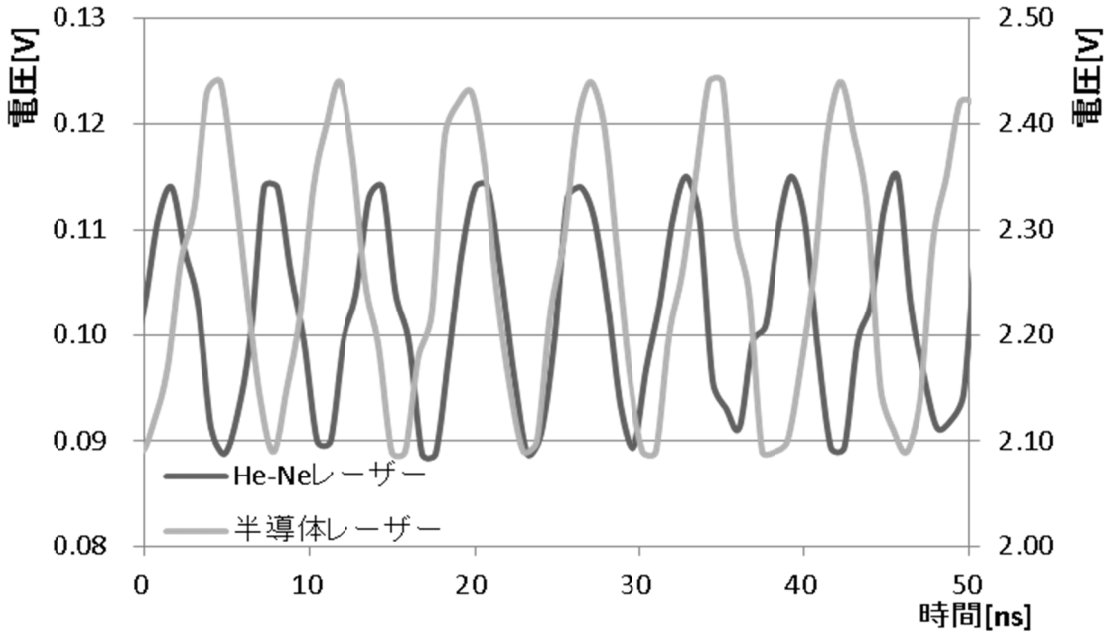

図 16　He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果の拡大図

He-Ne レーザー光が強め合った回数n_aは 14820 回で、半導体レーザー光が強め合った回数n_bは 14791 回だった。これらの値と He-Ne レーザー光の波長λ_a＝632.8nm を(4)式に代入し、半導体レーザー光の波長λ_bを求めた。

$$\lambda_b = \frac{632.8 * 14820}{14791}$$

$$= 634.04\text{nm}$$

半導体レーザー光の波長の公称値は 634.0nm であるから、ほぼ正確に波長を測定することができた。

3.3.6.2. He-Ne レーザー光と 785nm の半導体レーザー光での実験

634.0nm の半導体レーザー光の波長を求めることができたので、違う波長の半導体レーザー光の波長を求める。

図 17 に He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果を示す。

黒い波形は He-Ne レーザー光の波形を示し、灰色の波形は半導体レーザー光の波形を示している。波が大きく変化していることから、それぞれ干渉していることがわかる。

図 18 に He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果の拡大図を示す。

拡大図を見ると、He-Ne レーザー光は半導体レーザー光に比べて干渉波形の周期が短いことから、強め合った回数が多く、波長が短いことがわかる。

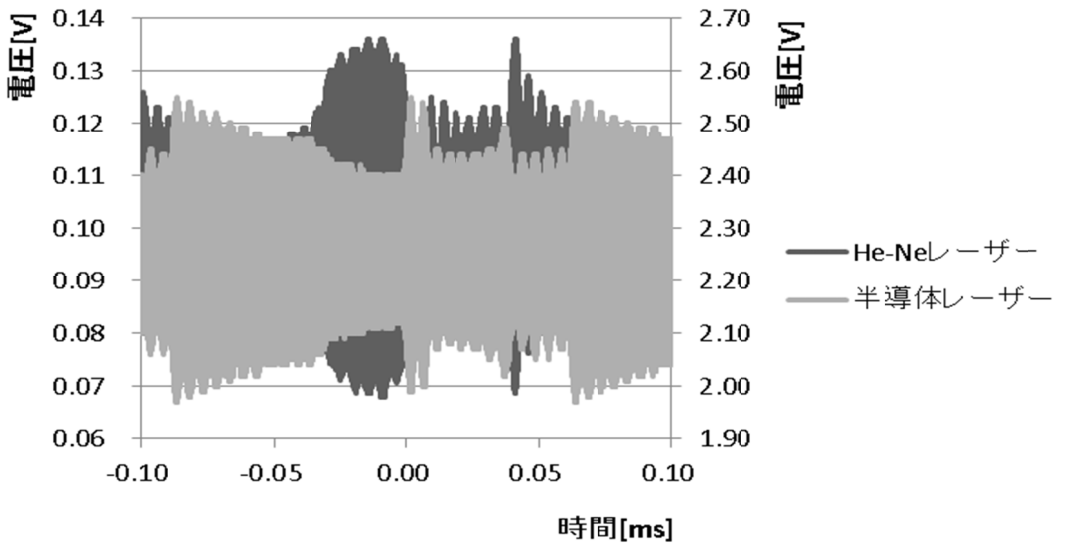

図 17　He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果

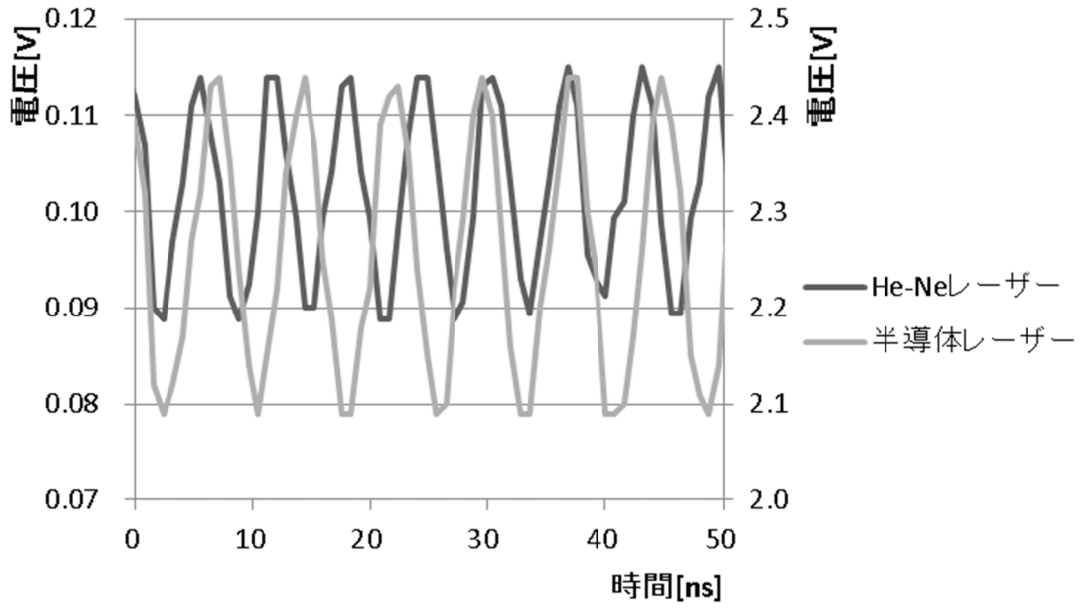

図 18　He-Ne レーザー光と半導体レーザー光をそれぞれ干渉させたときの測定結果の拡大図

拡大図を見ると、He-Ne レーザー光は半導体レーザー光に比べて周期が短いことから、強め合った回数が多く、波長が短いことがわかる。

He-Ne レーザー光が強め合った回数n_aは 24017 回で、半導体レーザー光が強め合った回数n_bは 19184 回だった。これらの値と He-Ne レーザー光の波長λ_a＝632.8nm を(4)式に代入し、半導体レーザー光の波長λ_bを求めた。

$$\lambda_b = \frac{632.8 * 24017}{19184}$$

$$= 792.2\text{nm}$$

半導体レーザー光の波長の公称値は 785nm であるから、約 7.2nm の誤差が生じた。

4. まとめ

干渉計を使って波長計を製作するため、様々な干渉計の特徴を調べた(p4 参照)。その中で、検出感度がよく、構成が簡単なマイケルソン干渉計を用いて波長計を製作することにした。次に、実際にマイケルソン干渉計を製作して He-Ne レーザー光を使用し、干渉縞ができるかを調べた。干渉縞ができたので、ミラーを 1mm 移動させて光を干渉させ、He-Ne レーザー光の波長を求めた。しかし、求めた波長が公称値と違っていた。

求めた波長と公称値が違っていた原因は、移動距離が 1mm ちょうどではなかったからだと考え、マイケルソン干渉計による波長計を製作した。この装置は二つの光を使用し、ボイスコイルモータで直角プリズムを移動させて光を干渉させる。ボイスコイルモータを移動させるため、マイコンボードを用いてプログラムで制御した。移動するときに振動が発生したので、回路中にコンデンサを挿入して振動を抑えた。この方法でも光が干渉するのか確かめるため、He-Ne レーザー光だけを使用し実験した。光が干渉したので、He-Ne レーザー光と半導体レーザー光(634.0nm)を使用し、半導体レーザー光の波長を求めた。誤差が 0.04nm だったことから、ほぼ正確に波長を測定することができた。

同じ装置を使って違う波長の半導体レーザー光(785.0nm)の波長を求めた。しかし、こちらは誤差が 7.2nm となり、634.0nm の半導体レーザー光で実験したときよりも大きくなった。これらの結果から、この実験にはまだ改善すべきところがあるとわかる。具体的には、光が強め合った回数を正確に数えることである。

5. 参考文献

(1)F. Gires, and P. Tournois (1964). “Interféromètre utilisable pour la compression d'impulsions lumineuses modulées en fréquence”. Comptes Rendus de l'Académie des Sciences de Paris 258: 6112–6115.

(2)THORLABS, ソーラボジャパン総合カタログ Vol21, ソーラボジャパン株式会社, 2011

6.付録

表1　使用機器

名称	型番	規格	製造元	台数
リニアガイド	SSEB16-230		MiSUMi	1
直角プリズム	PS911		THORLABS	2
ボイスコイル	AVM20-10			1
直流電源	DK-910		sunhayato	2
He-Neレーザー		632.8nm		1
半導体レーザー		634.0nm ， 785nm		2
スマートセンサ	ZX-LDA11-N 2M		オムロン	1
ハーフミラー			THORLABS	1
ミラー			THORLABS	2
ビームスプリッター			THORLABS	1
オシロスコープ				1
電解コンデンサ		20μF,1000μF		2
Arduino UNO			Arduino	1
バイポーラ形リニア集積回路	TA8429H/HQ		TOSHIBA	1
ディテクター			THORLABS	2

ピークの数 n を求めるプログラムを以下に示す。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
	FILE* fp;
	double x;
	int m/*=4*/;		/* 2m+1=9 点の平滑化幅 */
	printf("smoothing points = ");
	scanf("%d",&m);
	int sw = 2*m+1;
	printf("smoothing width = %d¥n" , sw);

	int PlotWidth1 = 2*m+1;
	double data[PlotWidth1];  /* 波形データ */

	int PlotWidth2 = 2*m+1;
	double DATA[PlotWidth2];  /* 波形データ */

	double gomi1;

	int n1=0;
```

```
int n2=0;


float ramuda=632.8;
double ramuda1=632.8/1000000;
double ramuda2;
int i,j;
fp = fopen("2.3.csv", "rb");


int ret=0;
if(fp != NULL)
{
        i=0;
        j=0;
        /* ","を考慮して 2 列目のデータを取得(csv は,区切り)*/
        while(EOF != fscanf(fp,"%lf,%lf,%lf",&gomi1,&data[i],&DATA[j]))
        {
                i++;
                if(PlotWidth1==i)
                {
                        n1 += TopCheck1(data,m);
                        i=0;
                }


                j++;
```

```
                if(PlotWidth2==j)
                {
                        n2 += TopCheck2(DATA,m);
                        j=0;
                }
        }
        /*for(i=0;i<PlotWidth;i++)
        {
                printf("data[%d]=%lf¥n",i,data[i]);
        }*/
        // printf("p=%lf,old_p=%lf¥n",p);
        printf("n1 = %d¥n",n1);
        printf("n2 = %d¥n",n2);

        //距離
        x=n1*ramuda1;
        //波長
        ramuda2=(n1*ramuda1*1000000)/n2;
}
else
{
        printf("File not found¥n");
        return 0;
}
```

```
	printf("x =    %lf[mm]¥n", x);
	printf("ramuda1 =    %lf[nm]¥n", ramuda);
	printf("ramuda2 =    %lf[nm]¥n", ramuda2);
	return 0;
}

//頂点があるとき 1,頂点がないとき 0 を返す関数
//引数 m は平滑化幅 W=2m+1 を満たす数値
int TopCheck1(double data[],int m)
{
	int i;
	static double old_p;/*  一つ前の微分係数  */
	double p;/*  微分係数  */

	p = 0;
	for( i= 1; i < m+1; i++ )
	{
		p += ( data[ m + i ] - data[ m - i ] ) * i;     /*  平滑化微分のための重み係数
演算  */
	}
	if( p < 0 && old_p > 0 )        /*  今の傾きが負で，前の傾きが正なら  */
	{
		old_p = p;
		return 1;
```

```
        }
        old_p = p;
        return 0;
}

int TopCheck2(double DATA[],int m)
{
        int j;
        static double old_p;/* 一つ前の微分係数 */
        double p;/* 微分係数 */

        p = 0;
        for( j= 1; j < m+1; j++ )
        {
                p += ( DATA[ m + j ] - DATA[ m - j ] ) * j;     /* 平滑化微分のための重み
係数演算 */
        }
        if( p < 0 && old_p > 0 )        /* 今の傾きが負で，前の傾きが正なら */
        {
                old_p = p;
                return 1;
        }
        old_p = p;
        return 0;
}
```