2003 年度　卒業研究レポート

常微分作用素の固有値問題の数値計算

横山　和正

2004 年 2 月 27 日

目 次

1　固有値問題の数値実験をするにあたって

一般化した固有値問題

$$\text{(EP)} \qquad \begin{cases} -[(pu')' + ru] = \lambda u \\ u(a) = u(b) = 0 \end{cases}$$

はp, rを条件に従い変えることで、固有値、固有関数は変化していく。
その変化の仕方から、固有値、固有関数に規則性はないか？
また、固有関数はp, rを変えることで、グラフが右によったり、左によったりするが、グラフをみただけで、p, rがどんな値かある程度推測できればおもしろいと思いこの実験をしてみた。

2　固有値問題（EP) の定理

(EP)
$$\begin{cases} -[(pu')' + ru] = \lambda u \\ u(a) = u(b) = 0 \end{cases}$$

係数 p, r は閉区間 $[a, b]$ において連続な実数値関数であり、$p \geq \delta > 0$ をみたす定数 δ が存在する。

2.1　(EP) の固有値はすべて実数である。

2.2　(EP) の任意の固有値に属する固有空間は 1 次元である。

2.3　(EP) の任意の固有値に属する固有関数のうち実数値のものがある。

2.4　(EP) の異なる固有値に属する固有関数は直交する。

3　目標

一般的な固有値問題の区間を $[0,1]$ とすると

$$\text{(EP)} \qquad \begin{cases} -[(pu')' + ru] = \lambda u \\ u(0) = u(1) = 0 \end{cases}$$

となる。また、一般的な波動方程式 (WE) も x の区間を $[0,1]$ とすると

$$\text{(WE)} \qquad \begin{cases} u_{tt} = (pu_x)_x + ru \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = f(x),\ \ u_t(x,0) = g(x) \end{cases}$$

となる。
このようにして数値実験を行う。

3.1　問題

(EP),(WE) の数値実験から p, r の値を $p \equiv 1$,$r \equiv 0$ 以外にした時、$p \equiv 1$,$r \equiv 0$ の時と固有値、固有関数はどう違うか？

3.2　復習

$p \equiv 1$ かつ $r \equiv 0$ の場合 (EP) を解くと

$$\text{(EP)} \qquad \begin{cases} \lambda_n = (n^2\,\pi^{\,2})(n = 1, 2, \dots) \\ u_n(x) = C \sin n\pi x \end{cases}$$

となり、固有値の平方根の比 $\sqrt{\dfrac{\lambda_n}{\lambda_1}} = n$ となることがわかる。
また、(WE) の解は

$$\text{(WE)} \qquad \begin{cases} u(x,t) = \displaystyle\sum_{n=1}^{\infty} (a_n \cos n\pi t + b_n \sin n\pi t) \sin n\pi x, \\ a_n = 2 \displaystyle\int_0^1 f(x) \sin n\pi x \ \ dx, \\ b_n = \dfrac{2}{n\pi} \displaystyle\int_0^1 g(x) \sin n\pi x \ \ dx, \end{cases}$$

$u_{(x,t)}$ の各項が各 t について、x について周期 $\dfrac{2}{n}$ の周期関数となっているので、全体も周期 2 の周期関数ということがわかる。

3.3 (EP) の数値実験で調べたいこと

1 λ_n を小さい方から n 番目の固有値として、固有値の平方根の比 $\sqrt{\dfrac{\lambda_n}{\lambda_1}}$ はどうなるか？

（$p \equiv 1$,$r \equiv 0$ の時は $\sqrt{\dfrac{\lambda_n}{\lambda_1}} = n$）

2 固有関数にはどんな違いがみられるか？

関数は無限個存在するので全てを調べることはできない。そこで、いくつかの p, r を用意して数値実験を行ってみる。

4　計算方法の紹介

4.1　(EP) の差分法について

p, r を一般的にしてしまうとわかりにくいので、最初は $p \equiv 1$,$r \equiv 0$ として説明する。

$$(EP)\begin{cases} -u'' = \lambda u \qquad (0 < x < 1) \\ u(0) = u(1) = 0 \end{cases}$$

を差分法を利用して近似的に解く。

1　区間 [0,1] を N 等分し、

$$h = \frac{1}{N}, x_i = ih \qquad (i = 0, 1, \ldots, N)$$

とおく。

2　x_i での関数 $u(x)$ の近似値を u_i と書く。
二階導関数を二階中心差分商で近似すると、

$$(1) \qquad \begin{cases} -\frac{1}{h^2}(u_{i-1} - 2u_i + u_{i+1}) = \lambda u_i \qquad (1 \leq i \leq N-1) \\ u_0 = u_N = 0 \end{cases}$$

3　(1) の式を行列、ベクトルを用いて表示すると、

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & O \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ O & & & -1 & 2 \end{bmatrix}\begin{Bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{Bmatrix} = \lambda \begin{Bmatrix} u_1 \\ \vdots \\ u_{N-1} \end{Bmatrix}$$

上式の行列を A とする。
A の固有値 λ、固有ベクトル $(u_1, \ldots, u_{N-1})^T$ をシフト法で求める。

⋆ちなみに、一般化した p, r を用いた場合、
（$-[(pu')' + ru] = \lambda u$ の場合）
$p_{i\pm 1/2} = p(x_i \pm \frac{h}{2})$, $\bar{p_i} = \frac{1}{2}(p_{i+1/2} + p_{i-1/2})$, $r_i = r(x_i)$ とおいた時

$$A = \frac{1}{h^2}\begin{bmatrix} 2\bar{p_1} + h^2 r_1 & -p_{3/2} & & & & & O \\ -p_{3/2} & 2\bar{p_2} + h^2 r_2 & -p_{5/2} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -p_{i-1/2} & 2\bar{p_i} + h^2 r_i & -p_{i+1/2} & & \\ & & & \cdot & \cdot & \cdot & \\ O & & & & & -p_{N-1-1/2} & \bar{p}_{N-1} + h^2 r_{N-1} \end{bmatrix}$$

となるので、同様のやり方で解く。

⋆ちなみにプログラムは次のページに載せておく。

```
/*
 * testx-4.c --- 菊地文雄・山本昌宏『微分方程式と計算機演習』山
海堂
 *              pp.125--127 のプログラムを C 言語に翻訳したもの
を
 *              -(p u')'+r u=λ u, u(0)=u(1)=0
 *              にした.
 */

#include <stdio.h>
#include <math.h>

#define NDIM (1000)

#include <stdlib.h>
#include <limits.h>

int number_p = 0;
int number_r = 0;

double sqr(double x) { return x * x; }

/* [0,1] 上の一様乱数 */
double drandom()
{
  return random() / (double) INT_MAX;
}

/* プロトタイプ宣言（定義はずっと後に出てくる）*/
void TRID(double A[], double B[], double C[], double F[], int N, int ID);
void INPR(double *X, double *Y, int N, double *S);
void shinv(double a[], double b[], double c[], double x[], double y[],
           double eps, double h, double SHIFT, int N, int M, double *Eval,
           int *NITER, int *NPIVOT);
```

```
void show_vector(int n, char *name, double x[])
{
  int i;
  for (i = 1; i <= n; i++)
    printf("%s[%d]=%f\n", name, i, x[i]);
}

double Pa, Pb, Ra, Rb, Rc, Oa, Ob, Oc, Od;
double Ha, Hb, Ia, Ib, Ic, Ja, Jb, Jc, Jd;

double P(double x){
  if (number_p == 0)
    return 1.0;
  else if (number_p == 1)
    return Pa * x + Pb;
  else if (number_p == 2)
    return Ra * x * x + Rb * x + Rc;
  else if (number_p == 3)
    return Oa * x * x * x + Ob * x * x + Oc * x + Od;
}
double R(double x){
  if (number_r == 0)
    return 0.0;
  else if (number_r == 1)
    return Ha*x+Hb;
  else if (number_r == 2)
    return Ia * x * x + Ib * x + Ic;
  else if (number_r == 3)
    return Ja * x * x * x - Jb * x * x + Jc * x + Jd;
}

int main(void)
{
  int N, M, i, k;
```

```
double h, h2, h3, x, SHIFT, EPS, EVAL, MINEIGEN;
int NITER, NPIVOT;
double *A, *B, *C, *U, *V;
char fname[128];
FILE *fp;
double pi = 4 * atan(1.0);

printf("INPUT : N=\n");
scanf("%d",&N);
if(N<=1)  exit(0);
A = malloc((N+1) * sizeof(double));
B = malloc((N+1) * sizeof(double));
C = malloc((N+1) * sizeof(double));
U = malloc((N+1) * sizeof(double));
V = malloc((N+1) * sizeof(double));
EPS = 1e-15; M = 100;

printf("0: P(x)=1.0,\n");
printf("1: P(x)=ax+b,\n");
printf("2: P(x)=ax^2+bx+c,\n");
printf("3: P(x)=ax^3+bx^2-cx+d \n");
printf("P(x)=");
scanf("%d", &number_p);
if (number_p == 1) {
  printf("a: "); scanf("%lf", &Pa);
  printf("b: "); scanf("%lf", &Pb);
}
else if (number_p == 2){
  printf("a: "); scanf("%lf", &Ra);
  printf("b: "); scanf("%lf", &Rb);
  printf("c: "); scanf("%lf", &Rc);
}
else if (number_p == 3){
  printf("a: "); scanf("%lf", &Oa);
  printf("b: "); scanf("%lf", &Ob);
```

```
  printf("c: "); scanf("%lf", &Oc);
  printf("d: "); scanf("%lf", &Od);
}

printf("0: R(x)=0,\n");
printf("1: R(x)=ax+b,\n");
printf("2: R(x)=ax^2+bx+c,\n");
printf("3: R(x)=ax^3+bx^2+cx+d \n");
printf("R(x)=");
scanf("%d", &number_r);
if (number_r == 1) {
  printf("a: "); scanf("%lf", &Ha);
  printf("b: "); scanf("%lf", &Hb);
}
else if (number_r == 2){
  printf("a: "); scanf("%lf", &Ia);
  printf("b: "); scanf("%lf", &Ib);
  printf("c: "); scanf("%lf", &Ic);
}
else if (number_r == 3){
  printf("a: "); scanf("%lf", &Ja);
  printf("b: "); scanf("%lf", &Jb);
  printf("c: "); scanf("%lf", &Jc);
  printf("d: "); scanf("%lf", &Jd);
}

printf("number p=%d\n", number_p);
printf("number_r=%d\n", number_r);

h = 1.0/N;
h2 = h*h;
h3 = h/2;

/* 計算結果をファイルに書き出す */
printf("file name: ");
```

```
  scanf("%s", fname);
  fp = fopen(fname, "w");
  fprintf(fp, "# N, EPS, NITER\n");
  fprintf(fp, "# %d %g %g %d\n", N, EPS, M);
  fprintf(fp, "# number_p=%d, number_r=%d\n",number_p,number_r);
  if (number_p == 9) {
    fprintf(fp, "# a=%f, b=%f\n", Pa, Pb);
  }
  for (k = 0; k < 5; k++) {
    if (k == 0) {
      SHIFT = 0;
    }
    else {
      SHIFT = sqr((k + 1.0) / k) * EVAL;
    }
    for(i = 1; i < N; i++){
      x = i * h;
      A[i] = -P(x-h3)/h2;
      B[i] = (P(x+h3)+P(x-h3)+h2*R(x))/h2-SHIFT;
      C[i] = -P(x+h3)/h2;
    }
    shinv(A,B,C,U,V,EPS,h,SHIFT,N-1,M,&EVAL,&NITER,&NPIVOT);
    printf("反復回数 = %3d  EPS = %12.4e SHIFT = %12.4e NPIVOT = %3d\n",
   NITER, EPS, SHIFT, NPIVOT);
    if (k == 0) {
      MINEIGEN = EVAL;
      printf("固有値 = %20.15f\n",EVAL);
      fprintf(fp, "# 固有値 = %20.15f\n",EVAL);
    }
    else {
      printf("固有値 = %20.15f (%f)\n",EVAL, sqrt(EVAL / MINEIGEN));
      fprintf(fp, "# 固有値 = %20.15f (%f)\n",EVAL, sqrt(EVAL / MINEIGEN));
    }
#ifdef OLD
    printf("** 節点での固有関数の値 **\n");
```

```
    for(i=0; i<5; i++)
      printf("  I     U(I)    ");
    printf("\n");
    for(i=1; i<N; i++){
      printf("%3d%12.4e ", i, U[i]);
      if(i%5==0)  printf("\n");
    }
    if((N-1)%5!=0)  printf("\n");
#endif
    for (i = 0; i <= N; i++)
      fprintf(fp, "%f %g\n", i * h, U[i]);
  }
  fclose(fp);
  return(0);
}

void shinv(double a[], double b[], double c[], double x[], double y[],
           double eps, double h, double SHIFT, int N, int M,
   double *Eval, int *NITER, int *NPIVOT)
{
  int i;
  double s,t,ENEW,ERROR;
  /* J=1;
     これは RND() に渡す種(seed)だったが、RND() の代りに drandom() を

     使うことになったので不要になった */
  for(i=1; i<=N; i++){
    x[i]=drandom();
  }
  *NITER = 0;
  *Eval = SHIFT;
  /* ITERATION */
  do {
    for (i=1; i<=N; i++){
      y[i]=x[i];
```

```
    }
    TRID(a, b, c, y, N, *NITER);
    INPR(x, y, N, &s);
    INPR(y, y, N, &t);
    ENEW=s/t+SHIFT;
    t=sqrt(h*t);
    for(i=1; i<=N; i++)
      x[i]=y[i]/t;
    ERROR=fabs((ENEW-*Eval)/ENEW);
    *Eval=ENEW;
    *NITER=*NITER+1;
#ifdef OLD
    printf("NITER=%d, Eval=%f, ERROR=%f\n", *NITER, *Eval, ERROR);
#endif
  }
  while (ERROR > eps && *NITER < M);
  *NPIVOT=0;
  for (i=1; i<=N; i++) {
    if (b[i]<0.0) *NPIVOT=*NPIVOT+1;
  }
}

void TRID(double A[], double B[], double C[], double F[], int N, int ID)
{
  int i;
  double AA;
  for(i=1; i< N; i++){
    AA=A[i+1]/B[i];
    if(ID==0) B[i+1]=B[i+1]-AA*C[i];
    F[i+1]=F[i+1]-AA*F[i];
  }
  F[N]=F[N]/B[N];
  for(i=N-1; i>=1; i--)
    F[i]=(F[i]-C[i]*F[i+1])/B[i];
}
```

```
void INPR(double *X, double *Y, int N, double *S)
{
  int i;
  *S=0.0;
  for (i=1; i<=N; i++){
    *S=*S+X[i]*Y[i];
  }
}
```

4.1.1 二階中心差分商

$u(x)$ は十分滑らかな関数とする。
$h > 0$ とすると、Taylor の公式により、

$$(2)\quad u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{6}u^{(3)}(x) + \frac{h^4}{24}u^{(4)}(x+\theta_1 h)$$

$$(3)\quad u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{6}u^{(3)}(x) + \frac{h^4}{24}u^{(4)}(x+\theta_2 h)$$

$\theta_1\theta_2$は $0 < \theta_1, \theta_2 < 1$
式 (2), (3) より、(2)+(3) をすると、

$$\frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = u''(x) + \frac{h^2}{24}u^{(4)}(x+\theta_1 h) + u^{(4)}(x-\theta_2 h)$$

したがって、の近似として次式を用いることができる。

$$u''(x) \fallingdotseq \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}$$

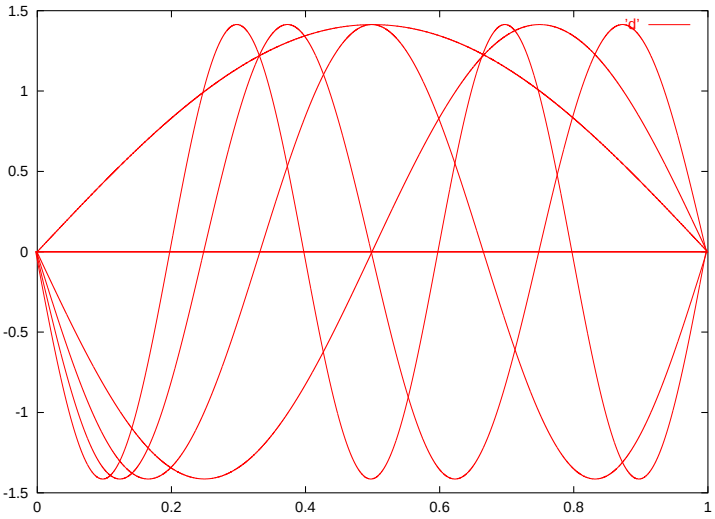
上式の右辺をの二階中心差分商という。

5　(EP) の実験結果

N は分割数であり、その下の数値が上から 1 番小さい方から 5 番目までの固有値 $\lambda_n(1 \leq n \leq 5)$、(　) の中は固有値の平方根の比 $\sqrt{\frac{\lambda_n}{\lambda_1}}$ である。また、その下のグラフが固有関数のグラフである。

5.1　$p \equiv 1$,$r \equiv 0$ の時

```
N=10000
    =     9.869604319806385
    =    39.478416307330299          (2.000000)
    =    88.826433027761311          (3.000000)
    =   157.913649644514152          (4.000000)
    =   246.740059277912309          (5.000000)
```



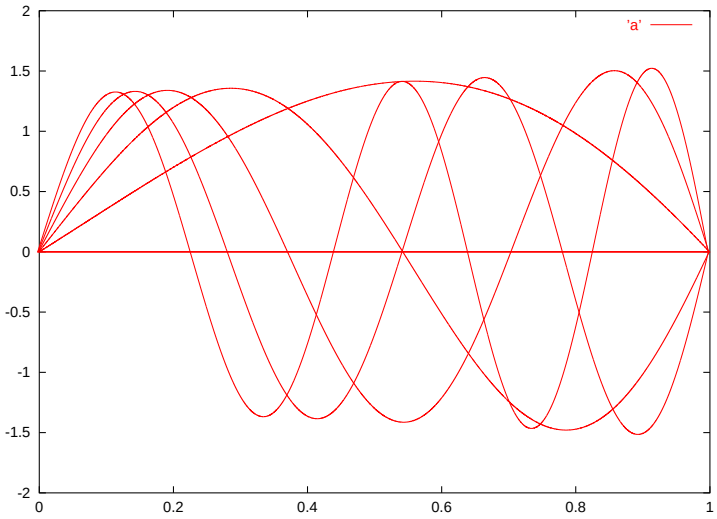
この結果はよく知られている。
結果からこのプログラムの精度は良いことがわかる。

5.2 $p(x)$ を一次関数とし、$r \equiv 0$

5.2.1 $p(x) = 2 - x, \quad r \equiv 0$ の時

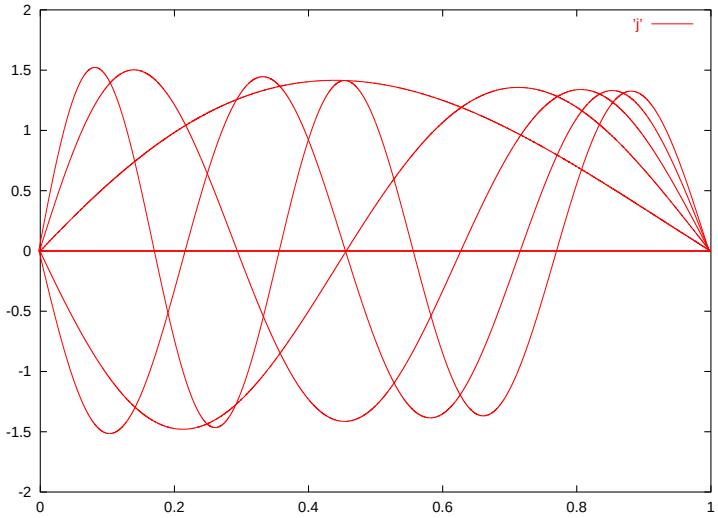
```
P(x)=2-x,              R(x)=0.0
N=1000
   =    14.337659518646349
   =    57.480093321256852          (2.002256)
   =   129.384528248992524          (3.004015)
   =   230.049833830313503          (4.005639)
   =   359.474933908855235          (5.007203)
N=10000
   =    14.337670654285057
   =    57.480282657210502          (2.002259)
   =   129.385497179565476          (3.004025)
   =   230.052907628521695          (4.005664)
   =   359.482451275960443          (5.007253)
N=100000
   =    14.337670729568003
   =    57.480284726616581          (2.002259)
   =   129.385507083067552          (3.004025)
   =   230.052938922273910          (4.005665)
   =   359.482526438896002          (5.007254)
```

$\sqrt{\frac{\lambda_n}{\lambda_1}}$ が n に近かったので、n に収束するのではないか？と思い、分割数を一桁ずつあげてみた。しかし、結果はみてのとおり n には収束しなかった。


'a'
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1

5.2.2 $p(x) = x + 1, \quad r \equiv 0$ の時

```
N=10000
    =    14.337670657309257
    =    57.480282662467125 (2.002259)
    =   129.385497183415680 (3.004025)
    =   230.052907630478558 (4.005664)
    =   359.482451277751181 (5.007253)
```

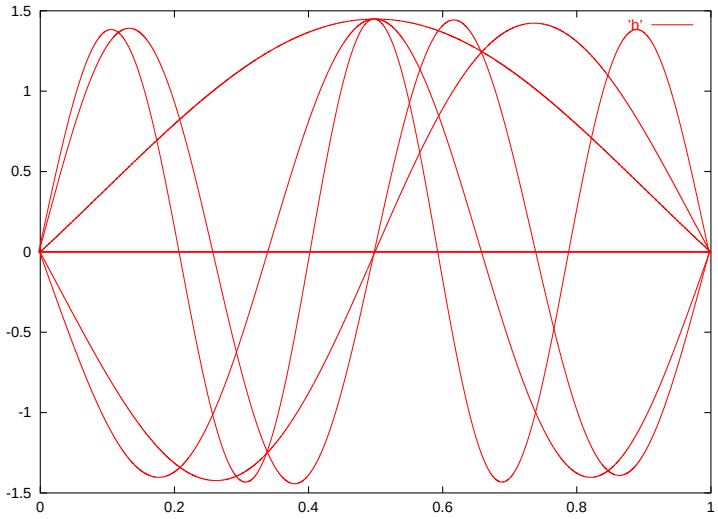


6.2.1 と 6.2.2 の固有値を比べると等しくなっていることがわかった。
どうやら、$p(x)$ が線対称の時等しくなるらしい？
また固有関数のグラフに変化がみられた。
$p(x)$ が一次関数の時、p が右下がりの時は固有関数は右によって、$p(x)$ が右上がりの時は固有関数は左によることがわかった。

5.3 $p(x)$ を二次関数とし、$r \equiv 0$

5.3.1 $p(x) = x^2 - x + 1, \quad r \equiv 0$ の時

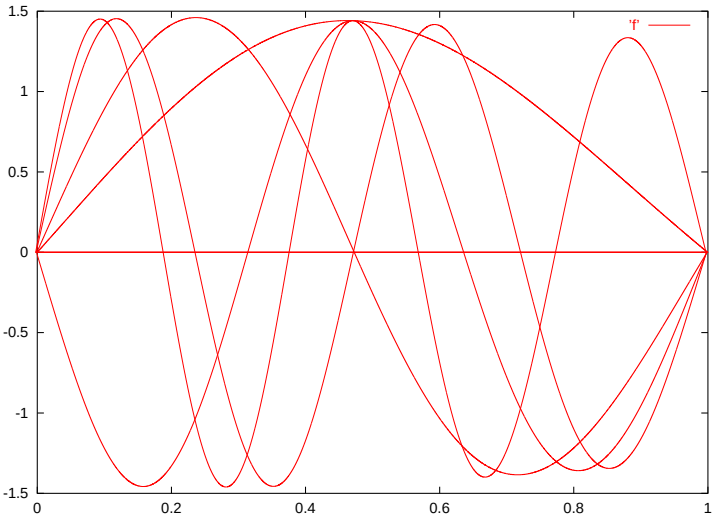
```
N=10000
    =     8.668037402234248
    =    33.189539689871808          (1.956772)
    =    74.074588438635388          (2.923306)
    =   131.315275158311181          (3.892219)
    =   204.910865488456722          (4.862080)
```



$p(x)$ を二次関数にしたら $\sqrt{\dfrac{\lambda_n}{\lambda_1}}$ は n から離れた。また、固有関数は得に変化がなかった。なお、ここでの $p(x)$ は $x = \dfrac{1}{2}$ で線対称になっている。

5.3.2 $p(x) = x^2 - \frac{1}{2}x + 1, \quad r \equiv 0$ の時

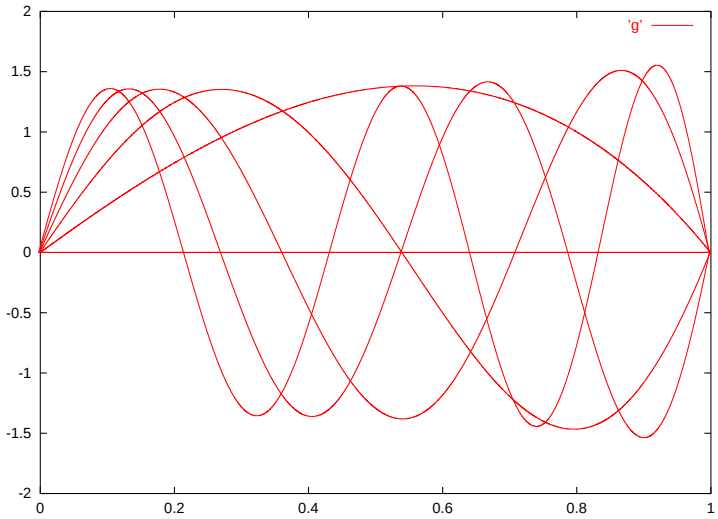
```
N=10000
    =    11.011702749917307
    =    42.595792145705204 (1.966781)
    =    95.246122046995467 (2.941009)
    =   168.957739679747931 (3.917074)
    =   263.730117833313159 (4.893872)
```



$p(x)$ の軸を左にずらしてみたら、固有関数のグラフが左によった。

5.3.3 $p(x) = -x^2 + \frac{1}{2}x + 1, \quad r \equiv 0$ の時

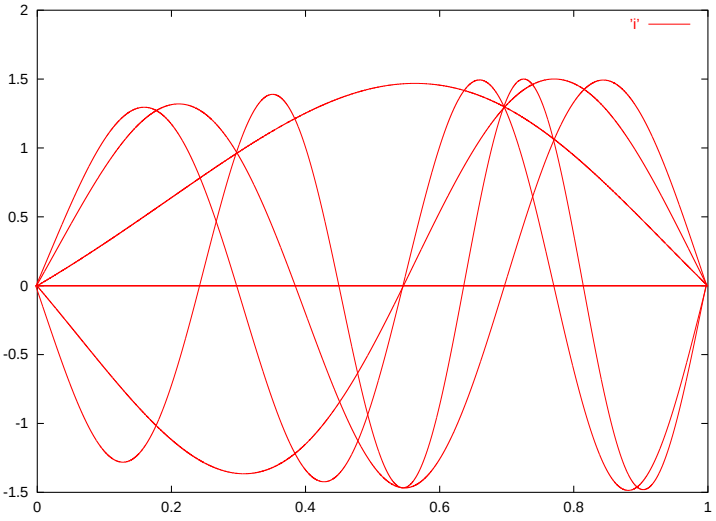
```
N=10000
    =     8.252007811964400
    =    34.596896060091552 (2.047570)
    =    78.528599311914689 (3.084850)
    =   140.037031160355667 (4.119473)
    =   219.120504618453793 (5.153018)
```



$p(x)$ の軸を左にずらして、符号を逆にしてみたら、固有関数のグラフが右によった。

5.3.4 $p(x) = x^2 - \frac{3}{2}x + 1, \quad r \equiv 0$ の時

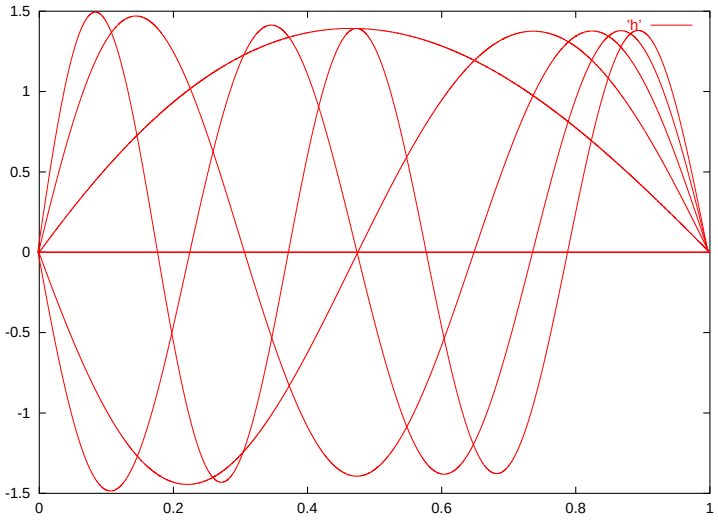
```
N=10000
    =     5.945493549552551
    =    22.364259753961043 (1.939471)
    =    49.744993617109010 (2.892549)
    =    88.079743735898504 (3.848966)
    =   137.367726142710609 (4.806715)
```



$p(x)$ の軸を右にずらしてみたら、固有関数のグラフが右によった。

5.3.5 $p(x) = -x^2 + \frac{3}{2}x + 1, \quad r \equiv 0$ の時

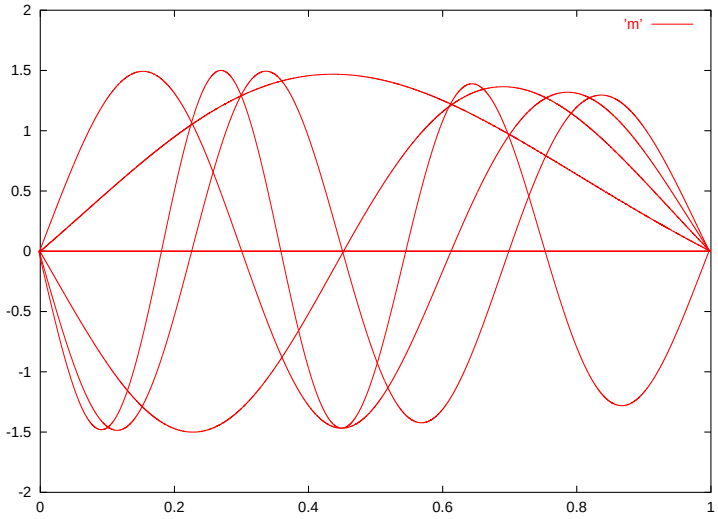
```
N=10000
    =    13.307074653841532
    =    54.779586090691204 (2.028935)
    =   123.913204055641160 (3.051529)
    =   220.702223968812632 (4.072509)
    =   345.145796759516543 (5.092840)
```



$p(x)$ の軸を右にずらして、符号を逆にしてみたら、固有関数のグラフが左によった。

5.3.6 $p(x) = x^2 - \frac{1}{2}x + \frac{1}{2}$, $r \equiv 0$ の時

```
N=10000
    =     5.945493549354590
    =    22.364259754002475 (1.939471)
    =    49.744993617859258 (2.892549)
    =    88.079743736468544 (3.848966)
    =   137.367726143282198 (4.806715)
```

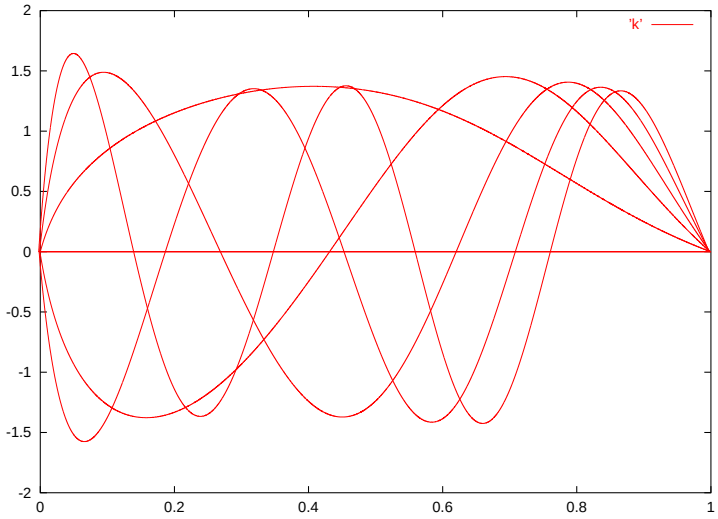


6.3.4 と 6.3.6 の固有値が等しくなった。
ここで使った $p(x) = x^2 - \frac{3}{2}x + 1$ と $p(x) = x^2 - \frac{1}{2}x + \frac{1}{2}$ は線対称である。

5.4 $p(x)$ を三次関数とし、$r \equiv 0$

5.4.1 $p(x) = 60x^3 - 90x^2 + 40x + 1, \quad r \equiv 0$ の時

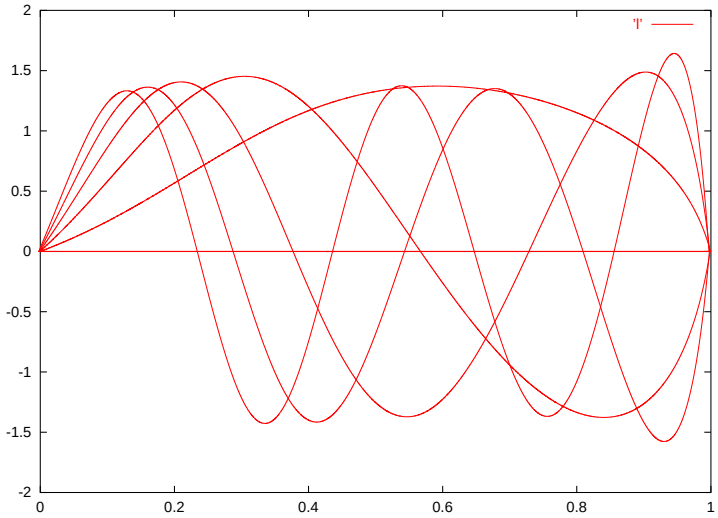
```
N=10000
    =    48.205437211125307
    =   212.985524188651794 (2.101973)
    =   485.107231267872010 (3.172275)
    =   867.196920478393281 (4.241416)
    =  1358.987996686991210 (5.309575)
```



この $p(x)$ は右上がりである。
固有関数は左によって、少し丸みがある。

5.4.2 $p(x) = -60x^3 + 90x^2 - 40x + 11, \quad r \equiv 0$ の時

```
N=10000
    =    48.205437211559676
    =   212.985524186940836 (2.101973)
    =   485.107231265713835 (3.172275)
    =   867.196920478054039 (4.241416)
    =  1358.987996685586495 (5.309575)
```

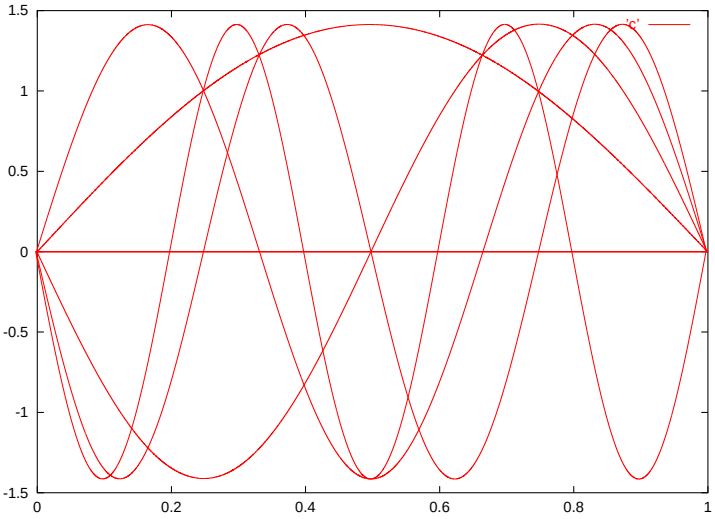


この $p(x)$ は右下がりである。
固有関数は右によって、少し丸みがある。

5.5 $p(x) \equiv 1$ として、$r(x)$ を変えてみる

5.5.1 $p(x) \equiv 1, \quad r(x) = x + 1$ の時

```
N=10000
  =    11.368507076400855
  =    40.978743486854093          (1.898574)
  =    90.326627963082302          (2.818748)
  =   159.413769029400896          (3.744650)
  =   248.240138590416848          (4.672876)
```

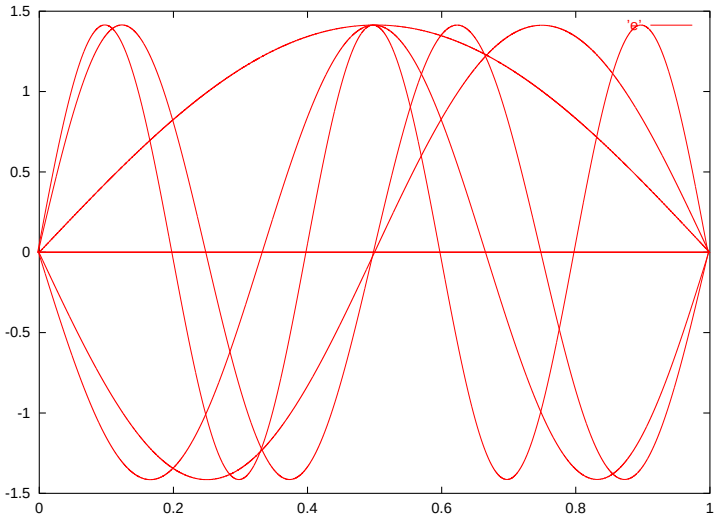


固有関数に特徴のある変化はない。

固有値の平方根の比 $\sqrt{\dfrac{\lambda_n}{\lambda_1}}$ は $r \equiv 0$ として $p(x)$ を関数にした時と比べると、n からのずれが大きい。

5.5.2 $p(x) \equiv 1, \quad r(x) = x^2 - \frac{3}{2}x + 1$ の時

```
N=10000
  =    10.401984903095805
  =    40.049148220864808          (1.962179)
  =    89.404189779552510          (2.931709)
  =   158.493851368271805          (3.903445)
  =   247.321390003306334          (4.876101)
```



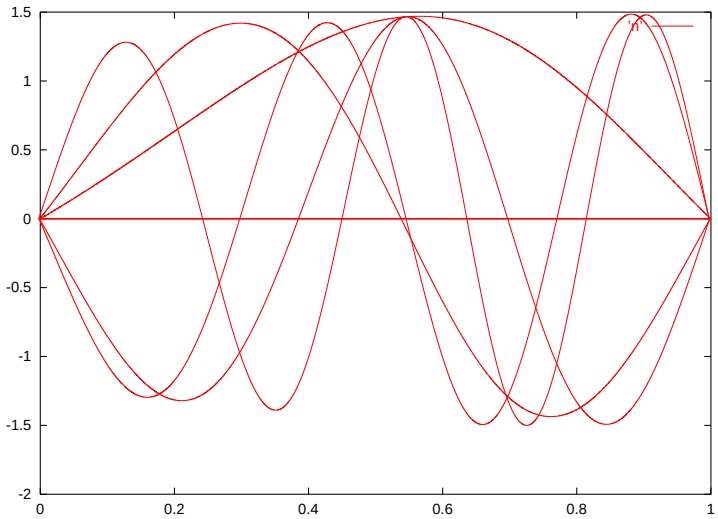
これも 6.5.1 の時と同じで、固有関数に特徴のある変化はない。

また、固有値の平方根の比 $\sqrt{\frac{\lambda_n}{\lambda_1}}$ は $r \equiv 0$ として $p(x)$ を関数にした時と比べると、n からのずれが大きい。

5.6 $p(x)$,$r(x)$ ともに関数を用いてみる

5.6.1 $p(x) = x^2 - \frac{3}{2}x + 1, \quad r(x) = 2x^3 + 3x^2 + x + 4$ の時

```
N=10000
    =     9.930721923979823
    =    26.455823417192573 (1.632188)
    =    53.738814007288447 (2.326235)
    =    92.073301932000732 (3.044924)
    =   141.361123688763968 (3.772894)
```



固有値の平方根の比 $\sqrt{\frac{\lambda_n}{\lambda_1}}$ が $r \equiv 0$ として $p(x)$ を関数にした時と比べると、n からのずれが大きい。
また、固有関数のグラフは右によっている。
（$p(x)$ の軸は右にずれている）

6　まとめ

6.1　固有値についてわかったこと

$r \equiv 0$ として、$p(x)$ を $p(x) = x^2 - \frac{3}{2}x + 1$ と $p(x) = x^2 - \frac{1}{2}x + \frac{1}{2}$ のように $x = 0$ または $x = 1$ を軸として線対称になるような関数を用いた場合、固有値は等しくなる。
また、$r(x)$ を関数にすると、固有値の平方根の比 $\sqrt{\frac{\lambda_n}{\lambda_1}}$ は $r \equiv 0$ として $p(x)$ を関数にした時と比べると、n から大幅にずれる。

6.2　固有関数についてわかったこと

$r(x)$ を変化させても固有関数のグラフは変化しない。
$p(x)$ が一次関数の時も二次関数の時も三次関数の時も、$p(x)$ のグラフが $\pm$ で右に下がってる時は固有関数は右によって、$p(x)$ のグラフが $\pm$ で右に上がってる時は固有関数は左による。
また、$p(x)$ が三次関数の時の固有関数のグラフは丸みがある。

参考文献

[1] 藤田宏，吉田耕作，現代解析入門，岩波書店 (1991).

[2] 菊池文雄，山本昌宏，微分方程式と計算機演習 (1991).