# n組の測定データを直線 で近似する場合、偏差の二乗和が最小になるような係数とおよび相関係数を求める式を要約した導出過程とともに示しなさい。 次に、この結果を利用して、与えられたn組の測定データから式(1)の係数およびおよび相関係数を求めるプログラムを作成し、次の表の10組のデータ（負荷応力と伸びの関係）に対するおよびおよび相関係数の値を求めなさい。ただし、キーボードから名前を指定したファイルからデータを読み込み、結果を式(1)のように表示すること。

## 導出

偏差の二乗和は次のようにあらわされる。

これが最小になるように係数とを決定する。つまり、

上部について、

下部について、

ここで、、をそれぞれ、とおくと、

これを上部の式に代入して、

について解くと、

ここで、、をそれぞれ、とおくと、

となり、、が決定できる。

　また、相関係数rはとして下式で表される。

作成したプログラムを次頁に示す。

プログラム1-1　que1.c

#include <stdio.h>

#include <math.h>

#define N 10

double x[N], y[N];

double a0, a1;

double cxy;

int main(void){

double mx=0,my=0;

double sxy=0,sx=0,sy=0;

double r;

int i;

char filename[20];

FILE \*fp;

printf("データを読み込むファイル名を入力してください。（20字以内)\n");

scanf("%s",filename);

fp=fopen(filename,"r");

for(i=0; i<N; i++){

fscanf(fp, "%lf %lf", &x[i], &y[i]);

}

fclose(fp);

printf("%3s %8s %8s\n","i", "x[i]", "y[i]");

for(i=0; i<N; i++){

printf("%3d %8.1lf %8.2lf\n",i, x[i], y[i]);

}

/\* 平均 \*/

for(i=0; i<N; i++){

mx += x[i];

my += y[i];

}

mx = mx/(double)N;

my = my/(double)N;

printf("mx=%lf\nmy=%lf\n",mx,my);

/\* 偏差 \*/

for(i=0; i<N; i++){

sx += pow(x[i]-mx, 2);

sy += pow(y[i]-my, 2);

sxy += (x[i]-mx)\*(y[i]-my);

}

printf("sx=%lf\nsy=%lf\nsxy=%lf\n",sx,sy,sxy);

/\* 回帰直線 \*/

a1=sxy/sx;

a0=my-a1\*mx;

printf("y = f(x) = %lf + %lf \* x\n",a0, a1);

/\* 相関係数 \*/

r=sxy/sqrt(sx\*sy);

printf("r=%lf",r);

return 0;

}

## 測定データへの適用

表1に示すデータについて回帰直線と相関係数を求める。

表 1　負荷応力と伸びの関係

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [N] | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
| [mm] | 0.12 | 0.23 | 0.36 | 0.46 | 0.58 | 0.68 | 0.83 | 0.92 | 1.04 | 1.15 |

結果は以下のようになった。

# 例題7.6および7.8のプログラムを変更して、台形公式とシンプソンの1/3公式によって与えられた積分を数値的に求める1つのプログラムを作成し、分割数について次の数値積分を実行して、解析解との相対誤差を求め、分割数と誤差について考察しなさい。ただし、分割数は配列変数に初期化して与えること。また、解析解は、自分で求めて、プログラムに関数として定義して、その値を求めること。なお、解析解の導出過程も示すこと。

以下に作成したプログラムを示す。

プログラム2-1　que2.c

#include <stdio.h>

#include <math.h>

#include "trape.h"

#include "simps.h"

int div[3]={10,30,50};

double prim\_f(double x){

return x; //2.1

//return log(x+sqrt(9+x\*x)); //2.2

}

double f(double x){

return x\*exp(x); //2.1

//return 1/sqrt(9+x\*x); //2.2

}

int main(void){

double a=0,b=1;

int i;

double st,ss,strue;

double rerr\_tr,rerr\_si;

strue=prim\_f(b)-prim\_f(a);

printf("%3s,%10s,%10s,%10s,%10s,%10s\n","div", "trape", "simps", "true", "rerr(tra)", "rerr(sim)");

for(i=0;i<3;i++){

st=trape(a,b,div[i],f);

ss=simps(a,b,div[i],f);

rerr\_tr=(st-strue)/strue\*100;

rerr\_si=(ss-strue)/strue\*100;

printf("%3d,%10lf,%10lf,%10lf,%10lf,%10lf\n",div[i], st, ss, strue, rerr\_tr, rerr\_si);

}

return 0;

}

プログラム2-2　trape.h

double trape(double a, double b, int n, double (\*func)(double)){

double s,x,h;

int i;

h=(b-a)/n;

s=0;

for(i=1;i<n;i++){

x=a+h\*i;

s+=(func)(x);

}

s=h/2.0\*((func)(a)+2\*s+(func)(b));

return s;

}

プログラム2-3　simps.h

double simps(double a, double b, int n, double (\*f)(double)){

double x,s,s1,s2,h;

int i;

s=s1=s2=0;

h=(b-a)/(2\*n);

for(i=1;i<2\*n;i+=2){

x=a+i\*h;

s1+=(f)(x);

}

for(i=2;i<2\*n;i+=2){

x=a+i\*h;

s2+=(f)(x);

}

s=h/3\*((f)(a)+4\*s1+2\*s2+(f)(b));

return s;

}

・解析解の導出過程

・結果

表2に各手法で求めた解の比較を示す。

表 2　数値積分解と解析解の比較

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| div | trape | simps | true | rerr(tra) | rerr(sim) |
| 10 | 1.003696 | 1.000000 | 1.000000 | 0.369604 | 0.000027 |
| 30 | 1.000411 | 1.000000 | 1.000000 | 0.041078 | 0.000000 |
| 50 | 1.000148 | 1.000000 | 1.000000 | 0.014788 | 0.000000 |

台形公式及びシンプソンの1/3公式による解を比較すると、同一分割数の場合でもシンプソンの1/3公式による解の方が解析解との相対誤差が小さいことがわかる。

また、数値積分解は分割数を大きくしていくと解析解との相対誤差が小さくなることがわかる。

・解析解の導出過程

・結果

表3に各手法で求めた解の比較を示す。

表 3　数値積分解と解析解の比較

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| div | trape | simps | true | rerr(tra) | rerr(sim) |
| 10 | 1.098186 | 1.098612 | 1.098612 | -0.038834 | -0.000001 |
| 30 | 1.098565 | 1.098612 | 1.098612 | -0.004315 | 0.000000 |
| 50 | 1.098595 | 1.098612 | 1.098612 | -0.001553 | 0.000000 |

2.1の場合と同様の傾向がみられる。

# 初期値問題の解を刻み幅h=1/4として、*x*=[0, 1]の区間でオイラー法および2次のルンゲ・クッタ法によって、手計算で求め、それぞれの解の誤差を解析解との相対誤差として求め、比較しなさい。ただし、解析解は自ら求め、その導出過程を示すこと。

## オイラー法

例えば、*x*=0.25のとき、

## 2次のルンゲ・クッタ法

例えば、*x*=0.25のとき、

## 解析解

与式を変形して、

まず、次の同次形微分方程式の一般解を求める。

次に、与式の特解を求める。

と仮定すると、

よって、

故に、与式の一般解は

ここで、だから、

よって、与式の解析解は

となる。

## 比較

表4に各手法で求めた解の比較を示す。

* *y*: 解析解
* *y1*: オイラー法による解
* *y2*: 2次のルンゲ・クッタ法による解
* *err1*: *y*に対する*y1*の相対誤差[%]
* *err2*: *y*に対する*y2*の相対誤差[%]

表 4　各手法による解の比較

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x* | *y* | *y1* | *y2* | *err1* | *err2* |
| 0.00 | 0 | 0 | 0 |  |  |
| 0.25 | -0.03403 | 0.00000 | -0.03125 | -100.0 | -8.2 |
| 0.50 | -0.1487 | -0.06250 | -0.1367 | -58.0 | -8.07 |
| 0.75 | -0.3670 | -0.20313 | -0.3354 | -44.7 | -8.60 |
| 1.00 | -0.7183 | -0.44141 | -0.6513 | -38.5 | -9.32 |

表4から、オイラー法による解よりも2次のルンゲ・クッタ法よる解の方が解析解に対する相対誤差が小さいことがわかる。

# クランク・ニコルソン法による熱伝導方程式の数値解法用プログラム「プログラム例2-9A」を変更し、「計算工学2プリント(7)」の式(9.12)～(9.14)で記述される熱伝導問題を問題2-80と同じ条件で解き、グラフ化しなさい。この結果と陽差分公式による「プログラム例2-8」で解いてグラフ化した結果と比較して、論じなさい。

次頁の図4.1、図4.2に陽差分公式とクランク・ニコルソン法による解のグラフを示す。

空間刻みは0.5、時間刻みは0.1とした。

図 4.1　陽差分公式による解

図 4.2　クランク・ニコルソン法による解

図4.1及び図4.2には大きな差異は見られない。

次に、空間刻みを0.5、時間刻みを0.5とした場合のグラフを図4.3、図4.4に示す。

図 4.3　陽差分公式による解

図 4.4　クランク・ニコルソン法による解

図4.3と図4.4はグラフの形状が大きく異なる。これは陽差分公式の収束安定条件が、

であり、空間刻みを0.5、時間刻みを0.5とした場合には

となり、収束安定条件を満たさないためである。

# 「計算工学2プリント(9)」の例題9.2.1に示した波動方程式の差分方程式式(9.20)をexcelによる表計算によって解きなさい。ただし、式(9.25)および(9.25)に示された境界条件に注意すること。この結果(c=9.0)とcの値をc=1.0、4.0と変化させた場合の解を求め、c=1.0、4.0、9.0に対する解を3次元グラフとして示し、それらの結果を比較して論じなさい。

以下に計算結果を示す。

図 5.1　c=1.0

図 5.2 c=4.0

図 5.3 c=9.0

グラフの比較から、cの値が大きくなるほど3次元グラフの表示が荒くなっていることがわかる。これは時間*t*の刻みkおよび空間*x*の刻みhを下式によって決定したためである。

今回はすべてのcの値について、としたので、c=1.0, 4.0, 9.0のそれぞれについて、h=0.1, 0.2, 0.3となり、cが大きいほど空間刻みが荒くなっている。