**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**Nội suy đường cong B-spline không đồng nhất**

Người hướng dẫn**: Thầy Nguyễn Tấn Khôi**

Sinh viên thực hiện**:**

**Tên sinh viên : Võ Trí Phúc**

**MSSV: 102200350**

**LỚP: 20TCLC\_Nhat2**

**Đà Nẵng, 06/2020**

MỤC LỤC

[MỤC LỤC](#_Toc37650459)

[MỞ ĐẦU](#_Toc37650461)

[1. TỔNG QUAN ĐỀ TÀI 1](#_Toc37650462)

[2. CƠ SỞ LÝ THUYẾT 1](#_Toc37650463)

[2.1. Ý tưởng 1](#_Toc37650464)

[2.2. Cơ sở lý thuyết](#_Toc37650465) 2

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN](#_Toc37650466) 7

[3.1. Phát biểu bài toán 1](#_Toc37650467)

[3.2. Cấu trúc dữ liệu 1](#_Toc37650468)

[3.3. Thuật toán 1](#_Toc37650469)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 1](#_Toc37650470)

[4.1. Tổ chức chương trình 1](#_Toc37650471)

[4.2. Ngôn ngữ cài đặt 1](#_Toc37650472)

[4.3. Kết quả 1](#_Toc37650473)

[4.3.1. Giao diện chính của chương trình 1](#_Toc37650474)

[4.3.2. Kết quả thực thi của chương trình 1](#_Toc37650475)

[4.3.3. Nhận xét đánh giá 1](#_Toc37650476)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 1](#_Toc37650477)

[5.1. Kết luận 1](#_Toc37650478)

[5.2. Hướng phát triển 1](#_Toc37650479)

[TÀI LIỆU THAM KHẢO 2](#_Toc37650480)

MỞ ĐẦU

Mục đích thực hiện đề tài: Nhằm cung cấp phương pháp, cách giải quyết vấn đề từ những dữ kiện có sẵn từ đó tính xấp xỉ chính xác đường cong. Góp phần phục vụ cho các công việc chuyên môn như đồ họa máy tính, nghiên cứu toán học,hình ảnh động…Ngoài ra nội suy đường cong tham số B-spline còn cho ta thấy những ưu điểm và khắc phục các nhược điểm của đường cong Bezier trong thực tiễn.

Mục tiêu, phạm vi và đối tượng nghiên cứu: Đề tài này dùng phương pháp nội suy toàn cục (bình phương tối thiểu) để tái tạo các yếu tố cấu thành nên đường cong tham số B-spline từ những điểm dữ liệu có sẵn với sai số cho phép.

Cấu trúc đồ án môn học:

* Tổng quan đề tài.
* Cơ sở lý thuyết.
* Tổ chức cấu trúc dữ liệu và thuật toán.
* Chương trình và kết quả.
* Kết luận và hướng phát triển.

# TỔNG QUAN ĐỀ TÀI

Đường cong B-Spline được coi là tổng quát của đường cong Bezier và do đó có nhiều điểm tương đồng với nó. Tuy nhiên, chúng có nhiều đặc tính mong muốn hơn so với đường cong Bezier. Đường cong B-Spline yêu cầu nhiều thông tin hơn như độ của đường cong và vectơ nút, và nói chung liên quan đến một lý thuyết phức tạp hơn đường cong Bezier.

Định nghĩa về đường cong B-Spline có thể hiểu như sau:

Một hàm spline bậc *p* là một đa thức riêng phần cấp *p* có các đạo hàm cấp *p* liên tục ở mỗi nút. Một đường cong *C*(*u*) có (*n +* 1) đỉnh điều khiển được xây dựng bằng cách sử dụng các hàm spline làm hàm trộn như sau :

* Tập các nút *u*0*, u*1*,..., um,* với và *ui ≤ ui+*1;
* là đa thức riêng phần liên tục trên mỗi đoạn cong [*ui, ui+*1].

Với mỗi đỉnh điều khiển *Pi* ta kết hợp nó với một hàm trộn tương ứng là 

Các đoạn cong từng phần này nối nhau trơn và liên tục tại các điểm nút, làm cho toàn bộ đường cong liên tục. Tập các đường cong như vậy được gọi là đường cong spline.

Cho trước một vector nút, có nhiều họ các hàm trộn sao cho chúng có thể phát sinh ra mọi đường cong spline được định nghĩa trên vector nút đó. Một trong các họ hàm trộn cho phép biểu diễn bất kì đường cong spline nào dưới dạng một công thức chung bằng cách sử dụng khung điều khiển phù hợp, được gọi là cơ sở cho spline. Đặc biệt họ hàm trộn có giá mang nhỏ nhất đó được gọi là hàm B-spline (B: Basis).

Để biểu diễn đường cong và mặt cong tham số phức tạp, người ta thường sử dụng đường và mặt cong B-spline.

Vậy, để biểu diễn đường và mặt cong B-spline không đồng nhất một cách hoàn thiện, ta cần suy ra các đỉnh điều khiển Pi và tập vector nút.

# CƠ SỞ LÝ THUYẾT

## Ý tưởng

Đề tài này trình bày phương pháp tái tạo đường và mặt cong tham số từ tập điểm dữ liệu cho trước với một độ chính xác cho phép. Để tái tạo đường cong và mặt cong tham số, ta cần xây dựng các vectơ tham số từ tập điểm dữ liệu, sau đó các vectơ nút được suy ra từ vectơ tham số và được sử dụng trong phương pháp tái tạo đường cong. Từ đấy áp dụng công thức, giải hệ phương trình ma trận để tìm tập đỉnh điều khiển.

## Cơ sở lý thuyết

Giả sử ta có (*n +* 1) điểm dữ liệu *D* = {*Di*} = {*D*0*, D*1*, D*2*,…, D*n}, đường cong B-spline *C*(*u*) được xây dựng từ các điểm dữ liệu *D* có thể theo một trong các cách sau:

B1: Thiết lập các tham số cho đường cong B-Spline nội suy các điểm dữ liệu {Di

B2: Xây dựng vector tham số

B3: Xây dựng vector nút từ vector tham số (xây dựng vector nút đồng nhất)

B4: Xấp xỉ toàn cục đương cong B-spline bằng phương pháp bình phương tối thiểu

B5: Kiểm tra và đánh giá đường cong tái tạo.

***Ví dụ*** *. Cho tập điểm D = {D0,.. ,D6}, n = 6 như bảng sau:*

|  |  |  |
| --- | --- | --- |
| *D* | *x* | *y* |
| *0* | *0.0* | *10.00* |
| *1* | *6.48* | *22.45* |
| *2* | *15.79* | *29.63* |
| *3* | *25.00* | *31.25* |
| *4* | *34.81* | *27.04* |
| *5* | *43.52* | *16.71* |
| *6* | *50.00* | *0.00* |

*Xây dựng đường cong B-spline xấp xỉ theo phương pháp bình phương tối thiểu.*

***Bước 1****: Thiết lập các tham số cho đường cong B-Spline nội suy các điểm dữ liệu {Di}*

*Để nội suy (6 + 1) điểm dữ liệu, ta chọn đường cong B-spline C(u) sẽ có:*

* *h = 4 (h + 1) đỉnh điều khiển *
* *Bậc p = 2 thỏa n > h ≥ p ≥ 1*

*Tiếp theo cần đi tính vectơ tham số T và vectơ nút U.*

***Bước 2****: Xây dựng véctơ tham số T từ tập điểm dữ liệu*

*Vectơ tham số T = {t0, t1, …, tn}.*

*Ta được T = { 0.0, 0.21, 0.32, 0.5, 0.68, 0.79, 1.0 }*

*Tham số t sẽ được sử dụng để tính giá trị các hàm Ni,p(t).*

***Bước 3:*** *Xây dựng véctơ nút U cho đường cong B-spline nội suy*

*Với số đỉnh h và bậc p đã xác định, ta xây dựng được vectơ nút U :*

*U = { 0 , 0 , 0 , 0.5 , 1, 1, 1}, bậc p = 2*

***Bước 4****: Xây dựng hệ phương trình xác định các đỉnh điều khiển {Pi}*

*Phương trình đường cong B-spline: *

*a) Xây dựng điều kiện điểm đầu và điểm cuối*

*Để đường cong đi qua điểm đầu tiên và điểm cuối cùng:*

*D0 = C(0) = P0*

*Dn = C(1) = Ph*

*Như vậy có {Pi} (i = 1,…,h − 1) đỉnh điều khiển chưa biết. Cần xác định các điểm này.*

*Các đỉnh còn lại được tính gần đúng theo phương pháp bình phương bé nhất sao cho nó xấp xỉ với (n+1) đỉnh cần tìm, công thức là:*

*b) Xây dựng hàm bình phương khoảng cách*

*Đặt Qk = , ta có f chính là hàm vô hướng của (h-1) giá trị biến P1,…,Pn-1. Ta dùng bình phương bé nhất để tối thiểu f*

*Đạo hàm f theo Pt. Ta nhận thấy hàm f là một parabolloid elliptic với các biến từ P1…Ph-1 . Do vậy khi đạo hàm f theo từng tham số Pt , tất cả các Qk và hàm cơ sở tại đó đều là hằng số và đạo hàm riêng của nó phải bằng 0*

*Cho biểu thức trên bằng 0, ta có:*

*Do có (h − 1) biến, giá trị g =1,…,h − 1 nên có (h − 1) phương trình.*

*Và:*

*d) Giải hệ phương trình để tìm các đỉnh điều khiển {Pi}*

*Tính {Pi} dựa theo công thức sau: (NT.N).P=Q để hàm f() xấp xỉ*

*Với NT là ma trận chuyển vị của N.*

*Dữ liệu đầu vào là các véc tơ s chiều thì:*

*N là một ma trận (n-1)x(h-1)*

*Q là véc tơ có (h-1).s điểm*

*P là ma trận có (h-1).s phần tử*

*\*Với t=0.21:*

*TH1: k = 0, i = 0... (n + p – k) = (3 + 2 0) = 5.*

*TH2: k = 1, i= 0..4.*

*=*

*=0.58*

*=*

*=0.42*

*=*

*TH3: k = 2, i= 0..3*

*==0.121*

*+ =0.57*

*=0.08*

*Tương tự với các tham số tiếp theo ta được ma trận*

Ma trận N là giá trị hàm cơ sở tính được khi truyền các tham số u vào.

*Tính*

*= 12.5*

*6.48*

*15.19*

*(NT.N).P=Q*

*N\*NT=*

*=>>> P*

***Bước 5****: Kiểm tra và đánh giá đường cong tái tạo.*

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

Input là một mảng 1 chiều chứa n+1 điểm dữ liệu theo tọa độ Oxy. Mỗi hai phần tử liên tiếp của mảng sẽ lưu trữ tọa độ của một điểm dữ liệu Di. Dữ liệu được lấy từ file hoặc nhập trực tiếp vào.

Output là tập đỉnh gồm h+1 đỉnh và vector nút gồm h+p+2 phần tử sao cho  
n > h ≥ p ≥ 1.

## Cấu trúc dữ liệu

Sử dụng mảng động để quản lý:

Tập tọa độ điểm đầu vào gồm 2\*n phần tử kiểu double: mỗi điểm sẽ lưu trữ tọa độ của mình bằng cách lấy mỗi 2 phần tử liên tiếp trong mảng (Ví dụ: Điểm đầu tiên sẽ lấy D[0], D[1], điểm tiếp theo sẽ lấy D[2], D[3],.. để lưu tọa độ của mình )

Vector nút gồm h+p+2 phần tử, mỗi phần tử là một số thuộc kiểu double.

Ma trận hàm cơ sở gồm 5\*n phần tử từ đỉnh thứ 2 đến đỉnh thứ (n-1), mỗi phần tử là một số thuộc kiểu double, dùng để lưu cả ma trận chuyển vị của nó

Tập đỉnh điều khiển đầu ra gồm 2\*h phần tử kiểu double, mỗi đỉnh sẽ lưu trữ tọa độ của mình bằng cách lấy mỗi 4 phần tử liên tiếp trong mảng (Ví dụ: Điểm đầu tiên sẽ lấy P[0], P[1], điểm tiếp theo sẽ lấy P[2], P[3],… để lưu tọa độ của mình).

## Thuật toán

\* Nhập giá trị cho các điểm dữ liệu: Lặp i=0 →2\*(n+1)-1, nhập từng tọa độ vào mảng

\* Xây dựng vector tham số T từ tập điểm dữ liệu:

- Tính d: Lặp k=1→n, tính d theo công thức tổng xích ma |Dk-Dk-1|

- Tính vector tham số theo công thức với k từ 1 tới n-1

- Gán t0 = 0, tn = 1

\* Xây dựng vector nút U cho đường cong B-spline nôi suy:

- Khởi tạo mảng u[] để chứa vector nút, nhập n,p,t,sum=0

- Lặp j=1→n-p

+ Lặp i=j→j+p-1

+ sum = sum + ti, u[j+p]=sum/p

+ Gán sum=0, tiếp tục vòng lặp

- Lặp i=n+1→n+p+1, gán u[i] = 1

\* Xây dựng hệ phương trình xác định các đỉnh điều khiển:

- Xây dựng hàm cơ sở

+ Nhập n, i, p, t, u[], khởi tạo 2 biến tich1, tich2

+ if(u[i+p]==u[i]) tich1=0;

else tich1=((t-u[i])/(u[i+p]-u[i])) \* hamcosoN(n,i,p-1,t,u);

if(u[i+p+1]==u[i+1]) tich2=0;

else tich2=((u[i+p+1]-t)/(u[i+p+1]-u[i+1]))\* hamcosoN(n,i+1,p-1,t,u);

if(p==0) {

if(t>=u[i] && t<u[i+1]) return 1;

else return 0;

} else {

return tich1+tich2;

}

- Tính ma trận N và xuất

+ Nhập n, p, N[][], t[], u[]

+ Gán N[0][0]=1; N[n][n]=1

+Gán 0 cho các phần tử mảng N hàng 0

+ Gán 0 cho các phần tử mảng N hàng n

+ Lặp i=0→n-1: trong vòng lặp trên tiếp tục lặp j=0→n: Gán N[i][j] bằng cách gọi hàm cơ sở

+ Xuất mt N

\* Tính hoành độ của tập đỉnh P (tung độ làm tương tự)

- Khởi tạo mảng a[][], gán giá trị N[][] vào mảng a[][], nếu là cột cuối thì gán giá trị là hoành độ của tập đỉnh P→ma trận a

- Biến đổi ma trận a

- Tìm nghiệm x: Lặp i=n→0:

+ Gán biến s=0

+ Lặp j=i+1→n, s=s+a[i][j]\*x[j];

+ x[i] = (a[i][n+1]-s) / a[i][i]

- Lưu giá trị mảng x vào những vị trí chẵn của mảng P (với y thì lưu vào vị trí lẻ)

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

Chương trình được chia nhỏ thành hàm main và nhiều hàm con khác:

- Hàm main

- Hàm nhập và xuất điểm dữ liệu

- Hàm tính vector tham số T

- Hàm tính vector nut U

- Tạo hàm cơ sở

- Tao ma trận N từ hàm cơ sở

- Hàm biến đổi ma trận

- Hàm tìm x,y

- Hàm xuất đỉnh P

## Ngôn ngữ cài đặt

Ngôn ngữ lập trình C

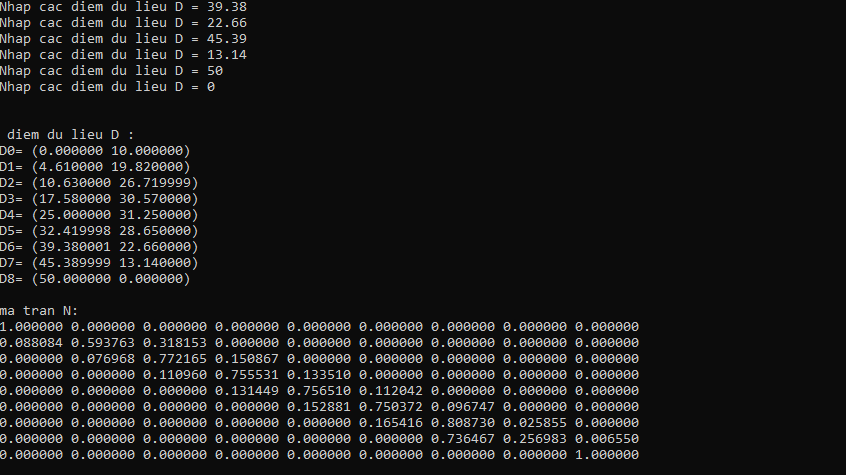
## Kết quả

### Giao diện chính của chương trình

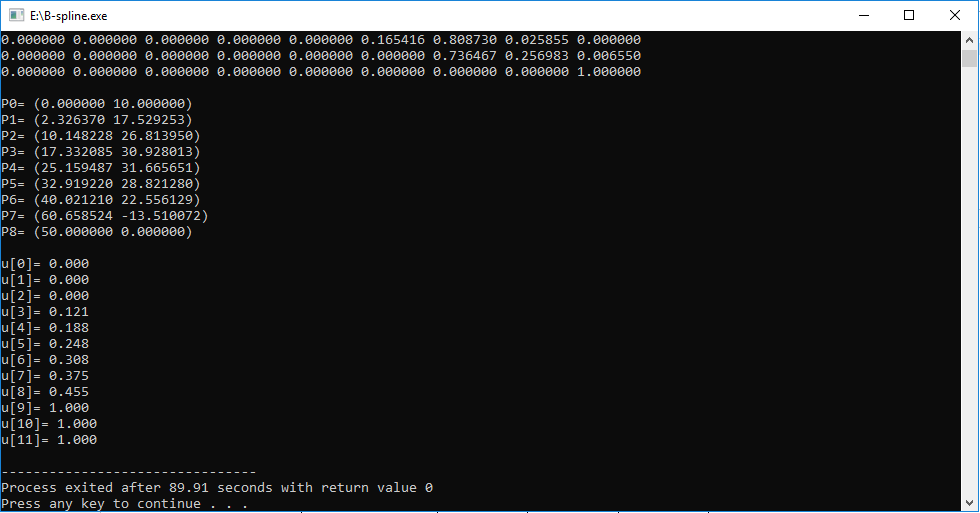
### Kết quả thực thi của chương trình

Nhập n, p và các tọa độ điểm dữ liệu từ bàn phím

Khi hoàn thành nhập, màn hình sẽ hiển thị kết quả ma trận N, tập đỉnh P và vector nút U



Ma trận N



Tập đỉnh P và vecor nút U

### Nhận xét đánh giá

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

KhoaCNTT-DHBKDaNang Tai tao duong cong Bézier - B-spline v3 B-spline non-uniform.

KhoaCNTT - DHBKDaNang Duong cong Bézier và B-spline v4.

Algorithm for Cubic Nonuniform B-Spline Curve Interpolation - Wolfram Demonstrations Project

PHỤ LỤC

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

#define EPS 0.01

/\*Thiet lap cac tham so cho duong cong B-spline noi suy cho cac diem du lieu\*/

void toadoD(int n, float D[100]) {

int z=0;

for(int i=0; i<=n; i++) {

printf ("D%i= (%f %f)\n", i, D[z], D[z+1]); //in ra toa do D

z+=2;

}

}

void toadoP(int n,float P[100]) {

int z=0;

for(int i=0; i<(n+1); i++) {

printf("P%i= (%f %f)\n",i,P[z],P[z+1]); //in ra toa do P

z+=2;

}

}

/\*Xay dung vecto tham so T tu tap diem du lieu\*/

void dodaidaycung(int n,float D[100],float \*d) {

int i=1;

for(int k=1; k<=n+1; k++) {

\*d+=sqrt(pow(D[i+1]-D[i-1],2.0)+pow(D[i+2]-D[i],2.0));

i+=2;

}

}

void T(int n,float D[100],float t[100],float d) {

t[0]=0; //t dau tien =0

t[n]=1; //t cuoi =0

int i=1;

for(int k=1; k<=n-1; k++) {

t[k]=t[k-1] + sqrt(pow(D[i+1]-D[i-1],2.0)+pow(D[i+2]-D[i],2.0))/d;

i+=2;

}

}

/\*xay dung vecto nut U cho duong cong B-spline noi suy\*/

void taovectonutU(int n,float t[100],int p,float u[100]) {

/\*tao vector nut U\*/

for(int i=0; i<=p; i++) {

u[i]=0;

}

double sum=0;

for(int j=1; j<=n-p; j++) {

for(int i=j; i<=j+p-1; i++)

sum+=t[i];

u[j+p]=sum/(float)p;

sum=0;

}

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

u[i]=1;

}

}

/\*Xay dung he phuong trinh xac dinh cac dinh dieu khien\*/

float hamcosoN(int n,int i,int p,float t,float u[100]) {

float tich1;

float tich2;

if(u[i+p]==u[i]) tich1=0;

else tich1=((t-u[i])/(u[i+p]-u[i])) \* hamcosoN(n,i,p-1,t,u);

if(u[i+p+1]==u[i+1]) tich2=0;

else tich2=((u[i+p+1]-t)/(u[i+p+1]-u[i+1]))\* hamcosoN(n,i+1,p-1,t,u);

if(p==0) {

if(t>=u[i] && t<u[i+1]) return 1;

else return 0;

} else {

return tich1+tich2;

}

}

void taomatranN(int n,int p,float N[100][100],float t[100],float u[100]) { //tinh mt N & xuat mt N

N[0][0]=1;

N[n][n]=1;

for(int j=1; j<=n; j++) N[0][j]=0; //gan 0 cho cac phan tu hang 0

for(int j=n-1; j>=0; j--) N[n][j]=0; //gan 0 cho cac phan tu hang n

for(int i=0; i<=n-1; i++)

for(int j=0; j<=n; j++) N[i][j] = hamcosoN(n,j,p,t[i],u); //tu hang 1->hang -1 tinh thong qua cosoN

for(int i=0; i<=n; i++) {

for(int j=0; j<=n; j++) printf("%f ",N[i][j]); //xuat mt N

printf("\n");

}

printf("\n");

}

void Dx(int n,float N[100][100],float a[100][100],float D[100]) { //gan N[][] cho a[][] va gan Dx cho a[i][n+1]

for(int i=0; i<=n; i++) {

for(int j=0; j<=n+1; j++) {

if(j==n+1) a[i][j]=D[2\*i];

else a[i][j]=N[i][j]; //neu la cot cuoi thi gan Dx vao,ko phai thi gan N[][]

}

}

}

void Dy(int n,float N[100][100],float a[100][100],float D[100]) { //gan N[][] cho a[][] va gan Dy cho a[i][n+1]

for(int i=0; i<=n; i++) {

for(int j=0; j<=n+1; j++) {

if(j==n+1) a[i][j]=D[2\*i+1];

else a[i][j]=N[i][j]; //neu la cot cuoi thi gan Dy vao,ko phai thi gan N[][]

}

}

}

void hoandoi(int n,float a[][100],int i,int j) {

printf("doi hang %i va %i",j,i);

float t;

for(int k=0; k<=n+1; k++) {

t=a[j][k];

a[j][k]=a[i][k];

a[i][k]=t;

}

}

void tamgiactren(int n,float a[][100]) {

int i,j;

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

if(a[i][i]==0)

for(int j=i+1; j<=n; j++)

if(a[j][i]!=0) {

hoandoi(n,a,i,j);

break; //dung

} else {

printf("");

return ;

}

float m;

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

m=-a[j][i]/a[i][i]; //tinh he so rut gon cua hang i+1 so voi i

for(int k=i; k<=n+1; k++) a[j][k]=a[j][k]+(a[i][k]\*m);

}

}

}

void nghiem(int n,float x[],float a[][100]) {

int i;

for (i = n; i >=0 ; i--) {

float s=0;

for (int j = i+1; j <=n; j++) s=s+a[i][j]\*x[j];

x[i] = (a[i][n+1]-s) / a[i][i]; // ( xi=hesotudohangi-CacTongCua(xk\*ak) ) / a[i][i]

//k LaViTriSauI->ViTriN

}

}

void LuuXVaoP(int n,float x[100],float P[100]) {

for(int i=0; i<=n; i++)

P[2\*i]=x[i]; //luu nghiem x vua tinh dc vao vi tri chan mang P

}

void LuuYVaoP(int n,float x[100],float P[100]) {

for(int i=0; i<=n; i++)

P[2\*i+1]=x[i]; //luu nghiem x vua tinh dc vao vi tri le mang P

}

int main() {

int n;

int p;

float d=0;

printf("Nhap n = ", n);

scanf("%d", &n);

printf("Nhap p = ", p);

scanf("%d", &p);

float t[100];

float u[100];

float D[100];

float P[100];

float N[100][100];

float a[100][100];

float r[100];

float x[100];

for(int i=0; i<2\*(n+1); i++) {

printf("Nhap cac diem du lieu D = ", D[i]);

scanf("%f",&D[i]);

}

printf("\n\n diem du lieu D :\n");

toadoD(n,D);

dodaidaycung(n,D,&d);

T(n,D,t,d);

taovectonutU(n,t,p,u);

printf("\n");

printf("ma tran N: \n");

taomatranN(n, p, N, t, u);

Dx(n,N,a,D);

tamgiactren(n, a);

nghiem(n, x, a);

LuuXVaoP(n,x,P);

Dy(n,N,a,D);

tamgiactren(n, a);

nghiem(n, x, a);

LuuYVaoP(n,x,P);

toadoP(n,P); //xuat P

printf("\n");

for (int i=0;i<=n+p+1;i++) { //xuat U

printf("u[%d]= %.3f",i,u[i]);

printf("\n");

}

return 0;

}