**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA: TOÁN – TIN HỌC**



**BÀI THU HOẠCH LỚP THỰC HÀNH**

**MÔN PHƯƠNG PHÁP TÍNH**

**Họ và tên sinh viên:** Nguyễn Hải Đăng

**Mã số sinh viên:** 20120049

**Giảng viên lý thuyết:** Thầy Vũ Đỗ Huy Cường

**Giảng viên thực hành:** Thầy Phạm Trương Hoàng Đức

**Lớp lý thuyết/Nhóm thực hành:** 20\_4/2

**Học kỳ - Niên khoá:** HK2 **-** 2021-2022

Thành phố Hồ Chí Minh, ngày 02 tháng 06 năm 2022

# BÀI 1:

1. Tạo ma trận A cỡ 5 × 5 có giá trị nguyên ngẫu nhiên nằm trong khoảng [0, 10].

function [r] = Random5x5 ()

r = randi([0 10],5,5);

end

----

Tham số đầu vào: Không có.

Giá trị trả về: Một ma trận 5x5 có giá trị nguyên ngẫu nhiên nằm trong khoảng [0, 10] (Hàm randi của matlab thực hiện chức năng này)

1. Cộng thêm b vào các phần tử ở cột 2 và 4 của ma trận A, gán kết quả cho B.

function [B] = CongB(A, b)

A(:,2) = A(:,2) + b;

A(:,4) = A(:,4) + b;

B = A;

end

----

Tham số đầu vào: Ma trận A, số b cần cộng.

Giá trị trả về: Ma trận B đã cộng thêm b vào cột 2 và 4.

1. Chuyển các bội số của a trong A thành số a, gán kết quả cho C.

function [C] = BoiSo(A, a)

[row, col] = size(A);

for i=1:row

for j=1:col

if (mod(A(i,j),a) == 0)

A(i,j) = a;

end

end

end

C = A;

end

----  
Tham số đầu vào: Ma trận A và số a.

Giá trị trả về: Ma trận C đã chuyển các bội số của a (các số chia hết cho a là bội số của a) thành a hết.

1. Tính nghịch đảo mọi phần tử khác 0 trong A, gán kết quả cho D.

function [D] = NghichDao(A)

[row, col] = size(A);

for i=1:row

for j=1:col

if (A(i,j) ~= 0)

A(i,j) = 1/A(i,j);

end

end

end

D = A;

end

----

Tham số đầu vào: Ma trận A.

Giá trị trả về: Ma trận D đã nghịch đảo các phần tử, nếu A(i,j) với i,j là hàng và cột trong ma trận A.

1. Chuyển các giá trị nhỏ hơn trung bình cộng của A thành giá trị 0, gán kết quả cho F.

function [F] = TBC(A)

[row, col] = size(A);

TrBinhCong = mean(A, 'all') %Tinh trung binh cong

for i=1:row

for j=1:col

if (A(i,j) < TrBinhCong)

A(i,j) = 0;

end

end

end

F = A;

end

----

Tham số đầu vào: Ma trận A.

Giá trị trả về: Ma trận F đã chuyển các giá trị nhỏ hơn trung bình cộng của A thành giá trị 0, đầu tiên ta dùng hàm mean() trong matlab để tính trung bình cộng các phần tử của ma trận. Sau đó duyệt từng phần tử của ma trận, phần tử nào nhỏ hơn trung bình cộng thì phần tử đó sẽ trở thành 0.

# BÀI 2:

1. Nhập vào vector x = [a, 2a, a − 10, b, 2b + 15, 3b].

function [x] = NhapVector(a, b)

x = [a, 2\*a, a - 10, b, 2\*b + 15, 3\*b];

end

----

Tham số đầu vào: 2 số a, b.

Giá trị trả về: Vector x = [a, 2a, a − 10, b, 2b + 15, 3b].

1. Cộng thêm 10 vào tất cả các phần tử của x gán kết quả cho y.

function [y] = CongThem10(x)

y = x + 10;

end

----

Tham số đầu vào: Vector x.

Giá trị trả về: Vector y sau khi cộng tất cả phần tử của x với 10.

1. Chuyển các bội số của 3 thành số 3, gán kết quả cho z.

function [z] = BoiSo3(x)

[row, col] = size(x);

for i=1:row

for j=1:col

if (mod(x(i,j),3) == 0)

x(i,j) = 3;

end

end

end

z = x;

end

----

Tham số đầu vào: Vector x.

Giá trị trả về: Vector z sau khi chuyển các bội số của 3 (là những số chia hết cho 3) thành số 3. Sử dụng hàm mod(a,b) để tính phần dư phép chia, trong đó a là số bị chia, b là số chia.

1. Gán cho vector v các giá trị lớn hơn 10 của x.

function [v] = GanV(x)

v = [];

[row, col] = size(x);

for i=1:row

for j=1:col

if (x(i,j) > 10)

v = [v,x(i,j)];

end

end

end

end

----

Tham số đầu vào: Vector x.

Giá trị trả về: Vector v có các giá trị lớn hơn 10 của x. Đầu tiên đặt v là vector rỗng, rồi dùng cú pháp v = [v, x(i,j)] với i,j là dòng và cột trong vector x.

1. Chuyển các giá trị nhỏ hơn trung bình cộng của x thành giá trị 0, gán kết quả cho w.

function [w] = TBC(x)

[row, col] = size(x);

TrBinhCong = mean(x, 'all')

for i=1:row

for j=1:col

if (x(i,j) < TrBinhCong)

x(i,j) = 0;

end

end

end

w = x;

end

----

Tham số đầu vào: Vector x.

Giá trị trả về: Ma trận w đã chuyển các giá trị nhỏ hơn trung bình cộng của A thành giá trị 0, đầu tiên ta dùng hàm mean() trong matlab để tính trung bình cộng các phần tử của ma trận. Sau đó duyệt từng phần tử của ma trận, phần tử nào nhỏ hơn trung bình cộng thì phần tử đó sẽ trở thành 0.

# BÀI 3:

1. DoThi.m

function [] = DoThi(f1,f2, x)

figure

plot(x,f1,'g',x,f2,'b--o')

end

----

Tham số đầu vào: Hàm f1, f2, x.

Không có giá trị trả về.

Trên màn hình sẽ xuất hiện 1 cửa sổ với 2 đồ thị f1, f2 màu khác nhau. Đồ thị f1 màu xanh lá cây, f2 màu xanh da trời.

1. File để chạy chương trình (TestCau3.m)

clear all; clc;

syms x;

x=0:pi/100:2\*pi;

f1=(x.^2).\*sin(x); %1

f2=(x.^3).\*cos(x); %2

DoThi(f1,f2,x)

# BÀI 4:

1. Hàm tính tiền taxi (TienTaxi.m)

function [r] = TienTaxi(a,b)

r = 0;

if (a > 0)

r = 14000;

end

if (a >= 2 && a <=25)

r = r + 16300\*(a-1);

elseif (a >= 26)

r = r + (16300\*24 + 13300\*(a-25));

end

r = r + 500\*b;

end

----

Tham số đầu vào: Quãng đường đã đi là a, thời gian chờ là b.

Giá trị trả về: Tiền taxi cần trả.

B1: Gán r = 0

B2: Nếu xe có đi thi giá r = 14000.

B3: Nếu số km từ 2 đến 25 thì r = r + 16300 \* (a – 1). Còn nếu như số km từ 26 trở lên thì r = r + (16300 \* 24 + 13300 \* (a – 25)).

B4: r = r + 500 \* b. Trả về r là tiền taxi cần trả. Kết thúc thuật toán.

# BÀI 5:

1. Đa thức Legendre

Ta đổi n ← n + 1, và thực hiện 1 số phép tính ta sẽ được:

Từ đó, ta có source code của đa thức như sau:

function [P] = Legendre(n, x)

if (n==0)

P = 1;

elseif (n==1)

P = x;

elseif (n==2)

P = (3\*(x^2)-1)/2;

else

P = ((2\*n-1)\*x\*Legendre(n-1,x)-(n-1)\*Legendre(n-2,x))/n;

end

end

----

Tham số đầu vào: Bậc đa thức n, biến x.

Giá trị trả về: Đa thức Legendre P.

Dùng hệ thức đệ quy với các trường hợp cơ sở n = 0, n = 1, n = 2 tương ứng.

1. Đa thức Chebyshev.

Ta đổi n ← n + 1, và thực hiện 1 số phép tính ta sẽ được:

Từ đó, ta có source code của đa thức như sau:

function [T] = Chebyshev(n,x)

if (n==0)

T=1;

elseif (n==1)

T=x;

else

T=2\*x\*Chebyshev(n-1,x)-Chebyshev(n-2,x);

end

end

----

Tham số đầu vào: Bậc đa thức n, biến x.

Giá trị trả về: Đa thức Chebyshev T.

Dùng hệ thức đệ quy với các trường hợp cơ sở n = 0, n = 1.

1. Dãy số Lucas.

Ta đổi n ← n + 2, và thực hiện 1 số phép tính ta sẽ được:

Từ đó, ta có source code của dãy số như sau:

function [L] = Lucas(n,p,q)

if (n == 0)

L=0;

elseif (n==1)

L=1;

else

L=p\*Lucas(n-1,p,q)-q\*Lucas(n-2,p,q);

end

----

Tham số đầu vào: n là số thứ tự của số cần tìm, biến x.

Giá trị trả về: Số thứ n của dãy số Lucas.

Dùng hệ thức đệ quy với các trường cơ sở n = 0, n = 1.

# BÀI 6:

Link bài làm:

<https://docs.google.com/forms/d/e/1FAIpQLSdL-NMUW0deUpGcqKubTgl7xRNTNvXYmvogWTR2psOO8Zoh-g/viewscore?viewscore=AE0zAgC8evOz6xVN0ztKF40in4E_KkhaYAXRm6KDGAzV7642LUuv46U1QMQ1M1gDaY6apiI>

1. Sai số tuyệt đối khi làm tròn số gần đúng.

Sai số tuyệt đối khi làm tròn số gần đúng = sai số khi làm tròn + sai số khi tính số gần đúng với sai số tương đối đã được cho trước.

1. Chữ số đáng tin.

B1: Tính sai số tương đối Δ.

B2: Nếu Δ ≤ 0.5\*10s thì s là chữ số đáng tin (nếu s là số nguyên âm thì s tương ứng với số thứ tự của chữ số sau dấu phẩy). Ngược lại s là chữ số đáng ngờ.

1. Công thức đánh giá sai số tổng quát.

Ta có hàm *f(x)*, khoảng phân ly nghiệm [a, b], nghiệm gần đúng *x\** được cho trước.

Sai số nhỏ nhất = với *t* là giá trị thuộc [a, b] sao cho *f(t)* là giá trị nhỏ nhất của *f(x)* trong đoạn [a, b].

1. Tính toán sai số.

Cho . Trong đó *f* khả vi liên tục theo các đối số *xi*. Khi đó sai số của *y* được xác định theo công thức sau:

* Sai số tuyệt đối: .
* Sai số tương đối: hay .

# BÀI 7:

1. Thuật toán phương pháp dây cung.

Diagram

Description automatically generated

1. Thuật toán phương pháp Seidel.

Diagram

Description automatically generated

1. Thuật toán phương pháp bình phương nhỏ nhất bằng đường thẳng.

Diagram, schematic

Description automatically generated

1. Thuật toán phương pháp bình phương nhỏ nhất bằng đường cong.

Diagram, schematic

Description automatically generated

# BÀI 8:

1. Sử dụng Matlab, tìm đa thức xấp xỉ của F(x) theo phương pháp đường cong Spline bậc 3.

Hàm tính đa thức xấp xỉ của F(x) theo phương pháp đường cong Spline bậc 3:

function [S1,S2,S3] = Spline(xx,yy)

h=xx(2:end)-xx(1:end-1);

VT=[1,0,0,0;h(1)/6,(h(1)+h(2))/3,h(2)/6,0;...

0,h(2)/6, (h(2)+h(3))/3,h(3)/6;0,0,0,1];

VP=[0;(yy(3)-yy(2))/h(2)-(yy(2)-yy(1))/h(1);...

(yy(4)-yy(3))/h(3)-(yy(3)-yy(2))/h(2);0];

m=inv(VT)\*VP;

M=yy(1:end-1)-m(1:end-1).\*h(1:end).^2/6

N=yy(2:end)-m(2:end).\*h(1:end).^2/6

syms x

S1=m(2)\*(x-xx(1))^3/6/h(1)+m(1)\*(xx(2)-x)^3/6/h(1)...

+M(1)\*(xx(2)-x)/h(1)+N(1)\*(x-xx(1))/h(1);

S2=m(3)\*(x-xx(2))^3/6/h(2)+m(2)\*(xx(3)-x)^3/6/h(2)...

+M(2)\*(xx(3)-x)/h(2)+N(2)\*(x-xx(2))/h(2);

S3=m(4)\*(x-xx(3))^3/6/h(3)+m(3)\*(xx(4)-x)^3/6/h(3)...

+M(3)\*(xx(4)-x)/h(3)+N(3)\*(x-xx(3))/h(3);

end

----

Ta sử dụng 4 giá trị: x0, x1, x2, x3 để tính toán đa thức Spline bậc 3.

Tham số đầu vào: xx và yy lần lượt là 2 vector x và F(x) tương ứng.

Giá trị trả về:

1. Sử dụng Matlab, tính W với các mốc giá trị trên bằng công thức Simpson 3/8.

Hàm tính tích phân W với x từ 0 tới 1.2 bằng công thức Simpson 3/8:

function [I1] = Simpson(xx,yy)

[xxRow,xxCol]=size(xx);

[yyRow,yyCol]=size(yy);

s=0;

for i = 4:3:xxCol

s=s+((xx(i)-xx(i-3))\*(yy(i)+3\*yy(i-1)+3\*yy(i-2)+yy(i-3))/8);

end

I1=s;

% rEI1=abs((I-I1)/I);

end

----

Tham số đầu vào: xx và yy lần lượt là 2 vector x và F(x) tương ứng.

Giá trị trả về: Giá trị W được tính bằng công thức Simpson 3/8.

Đầu tiên, ta tính số phần tử của xx và yy.

Sau đó, dùng vòng lặp bắt đầu từ i = 4, bước nhảy là 3 để tính tích phân theo công thức:

File TestCau8.m để test chương trình của Câu 8:

clear all; clc;

format short;

syms x;

F = 6\*pi\*(x^2)\*(6-x);

xx = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2];

yy = [double(subs(F,xx))];

[S1,S2,S3]=Spline(xx(:,1:4),yy(:,1:4))

W = Simpson38(xx,yy)

# BÀI 9:

1. Giải phương trình bằng phương pháp tiếp tuyến với ∆f = 10−5.

Hàm giải phương trình bằng phương pháp tiếp tuyến:

function [xBD]=TiepTuyen(f,x,xBD,delta)

k=1;

df=diff(f,x);

xBD=xBD-subs(f,x,xBD)/subs(df,x,xBD);

while abs(subs(f,x,xBD))>=delta

k=k+1;

xBD=xBD-subs(f,x,xBD)/subs(df,x,xBD);

end

end

----

Tham số đầu vào: hàm f với f(x) = 0, biến x, xBD là giá trị x ban đầu, delta là ∆f.

Giá trị trả về: xBD nghiệm của phương trình sau khi thực hiện phương pháp tiếp tuyến.

Tốc độ hội tụ của thuật toán nhanh. Tuy nhiên đạo hàm cấp 2 của hàm số phải không đổi dấu trên [a; b] thì thuật toán mới cho nghiệm hội tụ.

1. Tìm gần đúng y(1.1), y(1.2) bằng phương pháp Runge – Kutta bậc 3.

function [] = RungeKutta(f,a,b,h0)

xx=a:h0:b; h=xx(2:end)-xx(1:end-1);

[row,col]=size(xx);

x0=a;y=0\*xx;y0=1;

k1=h(1)\*f(x0,y0);

k2=h(1)\*f(x0+h(1)/2,y0+k1/2);

k3=h(1)\*f(x0+h(1),y0-k1+2\*k2);

y(1)=y0+(k1+4\*k2+k3)/6;

for i=2:col-1

k1=h(i)\*f(xx(i-1),y(i-1));

k2=h(i)\*f(xx(i-1)+h(i)/2,y(i-1)-k1+2\*k2);

k3=h(i)\*f(xx(i-1)+h(i),y(i-1)-k1+2\*k2);

y(i)=y(i-1)+(k1+4\*k2+k3)/6;

end

y

end

----

Tham số đầu vào: hàm f = y’ theo đề bài ra, a và b là [a; b] cần tính giá trị, h0 là khoảng cách giữa các số trong [a; b].

Giá trị trả về: là một vector [y(x + h0) y(x + 2\*h0) y(x + 3\*h0) … 0]

File TestCau9.m để test chương trình của Câu 9:

clear all; clc;

format short;

syms x1;

f1 = 2^x1 - 4\*x1; delta = 10^-5; xBD=3;

Nghiem = double(TiepTuyen(f1,x1,xBD,delta))

f2 = @(x,y) 0.15\*(x^2-y^2)\*cos(y);

y = RungeKutta(f2,1,1.2,0.1)

# BÀI 10:

1. Function cần điền là Công thức Runge-Kutta bậc 3 (Có thể điền là RungeKutta3)
2. Function cần điền là Công thức Runge-Kutta bậc 2 (Có thể điền là RungeKutta2)