EECS 639: Project 1

Gabrielle Gasser, Austin Bailey

Friday, September 22, 2017

Division of Labor Breakdown:

For parts A and B, both Austin and Gabrielle worked out the mathematical computation by hand and pair-programmed the solution.

For part C the manipulation of the equation was done by both parties until the solution was found by Austin. Austin also programmed the solution into the assignment.

For Part D the computation and programming was done by Gabrielle, with Austin assisting in the debugging phase.

Austin prepared the submission.

Eperimeter.m

function [C\_inner, C\_outer] = Eperimeter(a, b, n)

angles = gimmeAngles( n );

C\_inner = symsum(sqrt((a\*cos(angles(k)) - a\*cos(angles(k-1))).^2 + ( b\*sin(angles(k)) - b\*sin(angles(k-1))).^2 ), k, 2, n+1 );

[X\_Outer, Y\_Outer] = XY\_Outer( a, b, n, angles);

C\_outer = symsum(sqrt((X\_Outer(k) - X\_Outer(k-1)).^2 + (Y\_Outer(k) - Y\_Outer(k-1)).^2), k, 2, n+1);

MakeConic.m

function Z = MakeConic(A,B,C,D,E,F)

% Represents the conic defined by

%

% Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0

Z = struct('A',A,'B',B,'C',C,'D',D,'E',E,'F',F);

MakeEllipse.m

function Eta = MakeEllipse(Kappa)

phi = gimmiePhi(Kappa);

c=cos(phi)

s=sin(phi)

%%Assigns values for Aprime Cprime Dprime and Eprime for use later

Aprime = Kappa.A\*c\*c-Kappa.B\*c\*s+Kappa.C\*s\*s;

Cprime = Kappa.A\*s\*s+Kappa.B\*s\*c+Kappa.C\*c\*c;

Dprime = Kappa.D\*c - Kappa\*E\*s;

Eprime = Kappa.D\*s + Kappa.E\*c;

Fbar = -Kappa.F+(Dprime.^2/4\*Aprime)+(Eprime.^2/4\*Cprime);

%%checks that A' is > 0

if(Aprime<=0)

Eta = [];

end

%%checks that C' is > 0

if(Cprime<=0)

Eta = [];

end

%%checks that Fbar is >0

if(Fbar<=0)

Eta =[];

end

Eta.a = (sqrt(Fbar/Aprime));

Eta.b = (sqrt(Fbar/Cprime));

Eta.h = -(Dprime/2\*Aprime);

Eta.k = -(Eprime/2\*Cprime);

P1A.m

% Script P1A

%

% Plots the coefficients A, B, C, D, E, and F as the conic

%

% 2x^2 + xy + 3y^2 - x + y - 2

%

% is rotated around the origin (0,0).

close all;

n = 200;

Kappa = MakeConic(2,1,3,-1,1,-2);

phi = linspace(0,2\*pi,n)';

P = zeros(n,6);

% Store the A, B, C, D, E, and F values in the columns of P.

for k=1:n

Z = RotateConic(Kappa,phi(k));

P(k,:) = [Z.A Z.B Z.C Z.D Z.E Z.F];

end

% Plot the A, B, and C values

plot(phi,P(:,1),'-',phi,P(:,2),':',phi,P(:,3),'--')

xlabel('phi (the counter-clockwise rotation angle)')

legend('A','B','C');

title('Values of A, B, and C when 2x^2 + xy + 3y^2 - x + y - 2 is rotated \phi radians')

% Plot the D, E, and F values

figure;

plot(phi,P(:,4),'-',phi,P(:,5),':',phi,P(:,6),'--')

xlabel('phi (the counter-clockwise rotation angle)')

legend('D','E','F');

title('Values of D, E, F when 2x^2 + xy + 3y^2 - x + y - 2 is rotated \phi radians')

P1B.m

% Script P1B

clc

disp(' A B C D E F a b h k phi');

disp('-------------------------------------------------------');

for A = 0:1

for B = 0:2

for C = 0:2

for D = [-1 1]

for E = [-1 1]

for F = [-1 1]

Kappa = MakeConic(A,B,C,D,E,F);

Eta = MakeEllipse(Kappa);

if ~isempty(Eta)

a = Eta.a; b = Eta.b; h = Eta.h; k = Eta.k; phi = Eta.phi;

s1 = sprintf('%2d %2d %2d %2d %2d %2d',A,B,C,D,E,F);

s2 = sprintf('%6.3f %6.3f %6.3f %6.3f %6.3f',a,b,h,k,phi);

disp([s1 ' ' s2])

end

end

end

end

end

end

end

P1C.m

% Script P1C

% Illustrates ShowEllipse

close all

% Display the ellipse (x/3)^2 + (y/2)^2 = 1

% and what is obtained by rotating it counterclockwise pi/6, pi/3, and pi/2

% radians.

for i=1:4

subplot(2,2,i)

ShowEllipse(MakeEllipse(RotateConic(MakeConic(1/9,0,1/4,0,0,-1),(i-1)\*pi/6)))

title(sprintf('angle = %3d degrees',(i-1)\*30))

end

% Display 2x^2 + xy + 3y^2 - x + y - 2 = 0

figure

Eta = MakeEllipse(MakeConic(2,1,3,-1,1,-2));

ShowEllipse(Eta)

title(sprintf('a = %6.4f b = %6.4f (h,k) = (%6.4f,%6.4f) phi = %6.4f',...

Eta.a, Eta.b, Eta.h, Eta.k, Eta.phi))

P1D.m

% Script P1D

% Illustrates Eperimeter

[C\_inner, C\_outer] = Eperimeter(3, 2, 10000);

RotateConic.m

function New\_Kappa = RotateConic(Kappa,phi)

c = cos(phi);

s = sin(phi);

aa = Kappa.A\*(c.^2) - Kappa.B\*c\*s + Kappa.C\*(s.^2);

%bb = s\*(2\*Kappa.A\*c - Kappa.B\*s + 2\*Kappa.C\*c);

bb = 2\*Kappa.A\*c\*s + Kappa.B\*c.^2 - Kappa.B\*s.^2 - 2\*Kappa.C\*s\*c;

cc = Kappa.A\*s\*s + Kappa.B\*c\*s + Kappa.C\*c\*c;

dd = Kappa.D\*c - Kappa.E\*s;

ee = Kappa.D\*s + Kappa.E\*c;

ff = Kappa.F;

New\_Kappa = MakeConic(aa,bb,cc,dd,ee,ff);

end

ShowEllipse.m

function ShowEllipse(Eta)

%center = [Eta.h, Eta.k];

%verticalStrech = sqrt(Eta.a);

%$horizStrech = Eta.b;

%Plots a hundred data points in the x direction

%x = Eta.h-horizStrech:horizStrech/50:Eta.h+horizStrech;

%y=(sqrt(((x-Eta.h)/(Eta.a\*Eta.b)).^2))+Eta.k;

%{

t=linspace(0,2\*pi)

x=Eta.a\*cos(t);

y=Eta.b\*sin(t);

plot(x,y);

%}

x=(Eta.h-Eta.a:Eta.a/50:Eta.h+Eta.a);

y = Eta.b \* sqrt(1 - (((x-Eta.h)/Eta.a).^2) ) + Eta.k;

y2= -(Eta.b \* sqrt(1 - (((x - Eta.h)/Eta.a).^2) )) + Eta.k;

plot(x,y,'color','r'); hold on;

plot(x,y2,'color','b');

XY\_Outer.m

function [X\_outer, Y\_outer] = XY\_Outer( a, b, n, angles)

angles\_2 = angles( 2 : n+1 );

angles\_2(n+1) = angles(1);

X\_outer = a\* ((sin(angles) - sin(angles\_2))/(sin(angles)\*cos(angles\_2) - cos(angles)\*sin(angles\_2)));

Y\_outer = b\* ((cos(angles\_2) - cos(angles))/(sin(angles)\*cos(angles\_2) - cos(angles)\*sin(angles\_2)));

gimmieAngles.m

function angles = gimmeAngles( n )

x = n + 1;

angles = (2 \* pi \* ( i(1:1:x) - 1 ) ) / n;

gimmiePhi.m

function Kappa2 = gimmiePhi(Kappa)

Kappa2 = atan(Kappa.B/(Kappa.A-Kappa.C));