classdef set\_6

methods(Static)

% I had to tweak this function slightly. I noticed an issue with

% the discreatiztion of the fields. I noticed that each movement

% (delt\_a and delt\_n) were dependent on the step size of the search

% range construction. The Skel code had .01, meaning each inedx

% movement in a cardinal directioon is either +/- .01. So what

% happens if the user asks for a step size that is not an integer multiple of .01?

% This would imply that the mesh resolution needs to be increased such that there

% exists a natural number k that makes n\_k = 1/k \* (index # n) true

% where k is a real number that is less than step size used to

% construct the mesh grid

%% Problem 1:

%% Output:

% a\_best = 4.617187500000000

% n\_best = 1.517578125000000

function [a\_best,n\_best] = pattern\_fit(a0,n0,da,dn)

% finds the best fits of a model to included data using pattern search

% Some data I made up:

x\_data=[0, 0.0500, 0.1000, 0.1500, 0.2000, 0.2500, 0.3000, 0.3500, 0.4000, 0.4500, 0.5000, 0.5500, 0.6000, 0.6500, 0.7000, 0.7500, 0.8000, 0.8500, 0.9000, 0.9500, 1.0000];

z\_data=[0, 0.0288, 0.1095, 0.1945, 0.3290, 0.5470, 0.6779, 0.7737, 1.2348, 1.2512, 1.5704, 1.7016, 2.0294, 2.2959, 2.6138, 3.5973, 3.3025, 3.8515, 4.4374, 4.2621, 3.8980];

% We suspect this data will fit the model z=a\*x^n. You can plot and check,

% but comment this out before submitting.

% plot(x\_data,z\_data,'.','MarkerSize',12)

A=0:da:10; % This is the search range for "a".

N=0:dn:5; % This is the search range for "n".

[a\_fit,n\_fit]=meshgrid(A,N);

E = recalc\_e(da, 10, dn, 5, 0);

function E = recalc\_e(da, a, dn, n, v)

if ~v

A=0:da:a; % This is the search range for "a".

N=0:dn:n; % This is the search range for "n".

else

A= (a - da):da:(a + da); % This is the search range for "a".

N=(n - dn):dn:(n + dn); % This is the search range for "n".

end

[a\_fit,n\_fit]=meshgrid(A,N);

E=zeros(size(a\_fit));

for i=1:length(x\_data) % Sum the residual squared at each data point.

E = E + (z\_data(i)- a\_fit .\* x\_data(i) .^n\_fit) .^2;

end

end

% I find the locations of the arguments modularly and then use

% the result to place myself in E with row and col

i\_a = find( (a\_fit > a0 - da & a\_fit < a0 + da));

col = (i\_a(1) - mod(i\_a(1), length(N))) / length(N) + 1;

i\_n = find( (n\_fit > n0 - dn & n\_fit < n0 + dn));

row = mod(i\_n(1), length(N));

% E = z\_data - a\_fit \* (x\_data .^n\_fit) .^2;

% Comment this out before submitting, but you might be interested to see

% what the plot looks like re: the sum of residuals vs parameter pairs.

% It will also give you an idea of where the best fit is.

figure(1)

hold on

contour(a\_fit,n\_fit,E,min(E):1:max(E))

hold off

% Essentially, E is the function we are trying to minimize with respect to

% (a,n). Now you add your pattern search code below.

resolution = 0;

f\_nx = 0;

f\_x = 0;

f\_ny = 0;

f\_y = 0;

% This is where I make sure that the search does not access

% anyting outside the grid

while (resolution < 10 )

f\_0 = E(row, col);

if ( (row == 1) )

f\_x = E(row + 1, col);

f\_nx = abs(max(f)) + 1;

elseif (row == length(N))

f\_nx = E(row - 1, col);

f\_x = abs(max(f)) + 1;

else

f\_x = E(row + 1, col);

f\_nx = E(row - 1, col);

end

if ( (col == 1) )

f\_y = E(row, col + 1);

f\_ny = abs(max(f)) + 1;

elseif (col == length(A))

f\_ny = E(row, col - 1);

f\_y = abs(max(f)) + 1;

else

f\_y = E(row, col + 1);

f\_ny = E(row, col - 1);

end

f = [f\_0 f\_nx f\_x f\_ny f\_y];

% Where the pattern search makes its decsions

switch min(f)

case f\_0

da = da /2;

dn = dn / 2;

E = recalc\_e(da, A(col), dn, N(row), 1);

row = 2;

col = 2;

resolution = resolution + 1;

case f\_x

row = row + 1;

case f\_y

col = col + 1;

case f\_nx

row = row - 1;

case f\_ny

col = col - 1;

end

end

a\_best = A(col);

n\_best = N(row);

est = a\_best \* ( x\_data .^n\_best);

figure(2)

hold on

plot(x\_data,z\_data,'.','MarkerSize',12)

plot(x\_data, est, '--r');

hold off

% norm((est - z\_data), 2)

end

%% Problem 2:

%% Output:

% x\_max = 3.219380250507070

% y\_max = 1.583590903182648

function [f\_max, x\_max, y\_max]=univar3030(x0, y0, err\_a)

f =@(x,y) -3\*(1-x).^2.\*exp(-(x.^2) - (y+1).^2)- 5\*(x/5 - 3\*x.^3 - y.^5).\*exp(-0.4\*x.^2-y.^2) - 1/3\*exp(-(x+1).^2 - y^2);

xl = -3;

xu = 3;

yl = -3;

yu = 3;

err = 100;

dx = ((sqrt(5) - 1) / 2) \* (xu - xl);

dy = ((sqrt(5) - 1) / 2) \* (yu - yl);

x1 = xl + dx;

x2 = xu - dx;

y1 = yl + dy;

y2 = yu - dy;

while (err > err\_a)

x0 = x1;

y0 = y1;

if f(x1,y0) > f(x2, y0)

xl = x2;

x2 = x1;

x1 = xl + ((sqrt(5) - 1) / 2) \* (xu - xl);

else

xu = x1;

x1 = x2;

x2 = xu - ((sqrt(5) - 1) / 2) \* (xu - xl);

end

if f(x0,y1) > f(x0, y2)

yl = y2;

y2 = y1;

y1 = yl + ((sqrt(5) - 1) / 2) \* (yu - yl);

else

yu = y1;

y1 = y2;

y2 = yu - ((sqrt(5) - 1) / 2) \* (yu - yl);

end

if f(x1,y1) > f(x2, y2)

err = abs((1 - ((sqrt(5) - 1) / 2)) \* ( ((xu - xl) / x1)^2 + ((yu - yl) / y1)^2) ^ .5);

x\_max = x1;

y\_max = y1;

f\_max = f(x1, x2);

else

err = abs((1 - ((sqrt(5) - 1) / 2)) \* ( ((xu - xl) / x2)^2 + ((yu - yl) / y2)^2) ^ .5);

x\_max = x2;

y\_max = y2;

f\_max = f(x2, y2);

end

end

end

%% Problem 3:

%% Output:

% gmax = 0.236183273977815

% xmax = 0.500000007896745

% ymax = 0.500000000002345

function [gmax,xmax,ymax]=steepness(x0,y0,err\_a)

%tweaked golden Search

function [k\_max]=gold\_search(x0, y0, err\_a)

x = @(k) x0 + k \* gFx(x0, y0);

y = @(k) y0 + k \* gFy(x0,y0);

Fk = @(k) F(x(k), y(k));

klow = [(-x0 /gFx(x0, y0)) (-y0 /gFy(x0, y0))];

kupp = [((3 - x0) /gFx(x0, y0)) ((3 - y0) /gFy(x0, y0))];

[~, l] = min(abs(klow));

[ ~, u] = min(abs(kupp));

ku = klow(l);

kl = kupp(u);

if kl > ku

kt = kl;

kl = ku;

ku = kt;

end

err = 100000;

dk = ((sqrt(5) - 1) / 2) \* (ku - kl);

k1 = kl + dk;

k2 = ku - dk;

% k0 = k1;

while (err > err\_a)

if Fk(k1) > Fk(k2)

kl = k2;

k2 = k1;

k1 = kl + ((sqrt(5) - 1) / 2) \* (ku - kl);

else

ku = k1;

k1 = k2;

k2 = ku - ((sqrt(5) - 1) / 2) \* (ku - kl);

end

if Fk(k1) > Fk(k2)

err = abs((1 - ((sqrt(5) - 1) / 2)) \* ((ku - kl) / k1));

k\_max = k1;

else

err = abs((1 - ((sqrt(5) - 1) / 2)) \* ( ((ku - kl) / k2)));

k\_max = k2;

end

end

end

F = @(x,y) sqrt(x\*y) \* exp(- (x^2 + y));

gFx =@(x,y) exp(- (x^2 + y)) \* ( y - 4\* (x^2) \* y) / (2 \* (x\*y)^.5);

gFy = @(x,y) exp(- (x^2 + y)) \* ( x - 2 \* x \*y) / (2 \* (x\*y)^.5);

err\_x = 10;

err\_y = 10;

% This is the search alg

while (err\_x > err\_a && err\_y > err\_a)

k = gold\_search(x0, y0, err\_a);

fx = gFx(x0,y0);

fy = gFy(x0,y0);

err\_x = abs((k\* fx) / (x0 + k \* fx ));

err\_y = abs((k\* fy) / (y0 + k \* fy ));

x0 = x0 + k \* fx;

y0 = y0 + k \* fy;

end

gmax = F(x0, y0);

xmax = x0;

ymax = y0;

end

end

end