Matlab Appendix

Q12-13-14

```
%% Brachistochrone problem q.12
n    = 500;
g    = 9.8;
[X, Y] = meshgrid(1:n, 1:n);
n_y = 1./sqrt(2 * g * Y);


%% Q12

x0 = [1,   1];%xy
x1 = [300, 300];%xy
S0 = 0;
S  = runFSM(nref,x0,S0);

close all;
figure(1);
imagesc(S);axis square;

hold on; plot(x0(1),x0(2),'sb');
hold on; plot(x1(1),x1(2),'sk');

%% Q13
[Gx, Gy] = gradient(S);
alpha_k = 0.9;

%iterate
xOld = inf(length(x0'), 1);
x = x1';%starting from end point working back to initial position
path = x1';
maxIter = 10000;
tol = 1e-6;
f = @(x)(interp2(X, Y, S, x(1), x(2)));
for k = 1:maxIter,
    dx = -interp2(X, Y, Gx, x(1), x(2), 'linear', 0);
    dy = -interp2(X, Y, Gy, x(1), x(2), 'linear', 0);
    d = [dx; dy];

    if (norm(x-xOld)<tol*norm(xOld) || norm(x-x0') < 1 )
        break;
    end
    %constant step size
    alpha_k = 0.1 / norm(d);
    %update
    xOld = x;
    x = x + alpha_k*d;
    path = [path x];

    hold on; plot([xOld(1); x(1)], [xOld(2); x(2)], '-b',
'LineWidth', 2);

end

%% Q14 - analytic solution
%x = 0.5*k^2*(t - sin(t))
```

```
%y = 0.5*k^2*(1 - cos(t))
syms k t0 t1
%find k and t1
sol = solve(0.5*k^2*(t1 - sin(t1)) == x1(1), 0.5*k^2*(1 - cos(t1))
== x1(2));
kk = double(sol.k);

t = linspace(0, double(sol.t1), 1000);
x = 0.5*kk^2*(t - sin(t));
y = 0.5*kk^2*(1 - cos(t));

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

Q17

```
% q17
close all
clear all

load('I.mat');
load('mozart.mat');

eps = 1E-10;
F       = sqrt(1./I.^2 - 1 );
Fe    = F + eps * (F == 0);

x0 = [128,145];
z0 = 0;
z  = runFSM(Fe,x0,z0);

z11    = z(1,1);
z      = -z + z11;

figure;
surf(z);
colormap gray
shading interp;
axis('tight');
view(110,45);
axis('off');
camlight

figure;
surf(mozart);
colormap gray
shading interp;
axis('tight');
view(110,45);
axis('off');
camlight
```

Q18-19

```
%% Q18 - Surface normals
```

```
load Images.mat
load LightSources.mat
load mozart.mat

I = double(Images);
L = double(LightSources);
%N - number of pictures
[rows, cols, N] = size(I);
Image_n = zeros(rows, cols, 3);

Lpinv = pinv(L);
for n = 1:cols,
    for m = 1:rows,
        Imn = squeeze(I(m, n, :));
        Image_n(m, n, :) = Lpinv*Imn;
    end
end

%find p, q, N = (-p, -q, 1)
p = -Image_n(:, :, 1)./((Image_n(:, :, 3))  + eps);
q = -Image_n(:, :, 2)./((Image_n(:, :, 3))  + eps);


%% Q19 - Jacobi method
rows_p2 = rows + 2;
cols_p2 = cols + 2;
size = rows_p2*cols_p2;
R    = CreateDelOperators(rows_p2, cols_p2);%
dd   = -full(sum(R, 2));
D    = spdiags(dd, 0, size, size);
invD = spdiags(1./dd, 0, size, size);
[Dx, Dy] = CreateDerivativeOperators(rows_p2, cols_p2);

p = padarray(p, [1 1]);
q = padarray(q, [1 1]);
px = Dx*p(:);
qy = Dy*q(:);

A = R + D;
b = px + qy;

x0 = zeros(size, 1);
k_max = 50000;
tol = 1e-1;

x = x0;
k = 1;
% x = A\b;
while( norm(A*x-b) > tol && k <= k_max )
    x = invD*(b-R*x);
    k = k + 1;
end

Z = reshape(x, [rows_p2 cols_p2]);

figure;
colormap gray;
surf(Z,'FaceColor','interp',...
   'EdgeColor','none')
axis tight
shading interp
```

```
view(110,45)
camlight left;
axis off
title('Estimated Depth image')

figure;
colormap gray;
surf(mozart,'FaceColor','interp',...
   'EdgeColor','none',...
   'FaceLighting','gouraud')
axis tight
shading interp
view(110,45)
camlight left;
axis off
title('Ground Truth')
```

```
%% Q20

Q17
Q18

close all;
figure;
subplot(1, 2, 1);
surf(z);
colormap gray
shading interp;
axis('tight');
view(110,45);
axis('off');
camlight
title('|\nablaz|=F(x,y)')
subplot(1, 2, 2);
surf(Z);
colormap gray
shading interp;
axis('tight');
view(110,45);
axis('off');
camlight
title('|\nabla^2z|=p_x + q_y')
```