```
%-----------------------------------------------------%
% AFIT Qualifying Exam Question #4, 01 Dec 2014      %
% Timothy Coon                                        %
% Advisor: Dr Cobb
%-----------------------------------------------------%
% The problem solved here is given as follows:       %
% Minimize effort to get to the road subject to the %
% dynamic constraints (x is ind. var.)               %
%    dy = (dy/dx)*dx                                  %
% and the boundary conditions                         %
%    x0 = 0, y(0) = 1                                 %
%    xf = 3, y(xf) = FREE                             %
%-----------------------------------------------------%

% x boundary conditions
x0 = 0;                        % (l) force start at time = zero
xf = 3;                        % (l) fixed simulation time

% state limits
ymin = 0; ymax = 3;          % (l) min/max y-position

% control limits
umin = -10; umax = 10;         % (-) min/max slope

% state boundary conditions
y0 = 1;                       % (l)

%-------------------------------------------------------------------------%
%----------------------- Setup for Problem Bounds ------------------------%
%-------------------------------------------------------------------------%
iphase = 1;
% time = x-position
bounds.phase.initialtime.lower = x0;
bounds.phase.initialtime.upper = x0;
bounds.phase.finaltime.lower = xf;
bounds.phase.finaltime.upper = xf;

% states
% fixed initial state
bounds.phase.initialstate.lower = [y0];
bounds.phase.initialstate.upper = [y0];
% bounds for states after initial
bounds.phase.state.lower = [ymin];
bounds.phase.state.upper = [ymax];
% free final state
bounds.phase.finalstate.lower = [ymin];
bounds.phase.finalstate.upper = [ymax];

% control limits
bounds.phase.control.lower = [umin];
bounds.phase.control.upper = [umax];

% path constraints
% The integral constraints are inequalities. Express this by setting the
% lower bound to be the minimum required, then a guess for the upper
% bound which is NOT ARBITRARILY HIGH, MUST BE "CLOSE".
bounds.phase.integral.lower = [0];
bounds.phase.integral.upper = [100];
```

```
%-------------------------------------------------------------------------%
%----------------------- Provide Guess of Solution -----------------------%
%-------------------------------------------------------------------------%
guess.phase.time    = [x0; xf];
guess.phase.state   = [y0;y0];
guess.phase.control = [0; 0];
% the integral guess is to be a row vector (1xnd)
guess.phase.integral = [0];

%-------------------------------------------------------------------------%
%------------- Assemble Information into Problem Structure ---------------%
%-------------------------------------------------------------------------%
setup.name = 'Q4-Problem';
setup.functions.continuous = @Q4_Continuous;
setup.functions.endpoint = @Q4_Endpoint;
% setup.auxdata = auxdata;
setup.bounds = bounds;
setup.guess = guess;

setup.derivatives.supplier = 'sparseCD';
setup.derivatives.derivativelevel = 'second';
setup.derivatives.dependencies = 'sparseNaN';

setup.scales.method = 'none';

setup.method = 'RPM-Differentiation';

setup.mesh.method = 'hp-PattersonRao';
setup.mesh.tolerance = 1e-4;
setup.mesh.maxiteration = 10;

setup.mesh.phase.fraction = (1/n_mesh)*ones(1,n_mesh);
setup.mesh.phase.colpoints = n_cols*ones(1,n_mesh);

setup.nlp.solver = 'ipopt';
setup.nlp.ipoptoptions.linear_solver = 'mumps';
setup.nlp.ipoptoptions.tolerance = 1e-4;
setup.nlp.ipoptoptions.maxiterations = 100;

setup.displaylevel = 2;

%-------------------------------------------------------------------------%
%------------------------- Solve Problem Using GPOPS2 --------------------%
%-------------------------------------------------------------------------%
output = gpops2(setup);
output.result.nlptime
solution = output.result.solution;
GPOPS2_ExitFlags(output)

%% ----------------------------------------------------------------------%
%------------------------------- Plot Solution ---------------------------%
%-------------------------------------------------------------------------%
```

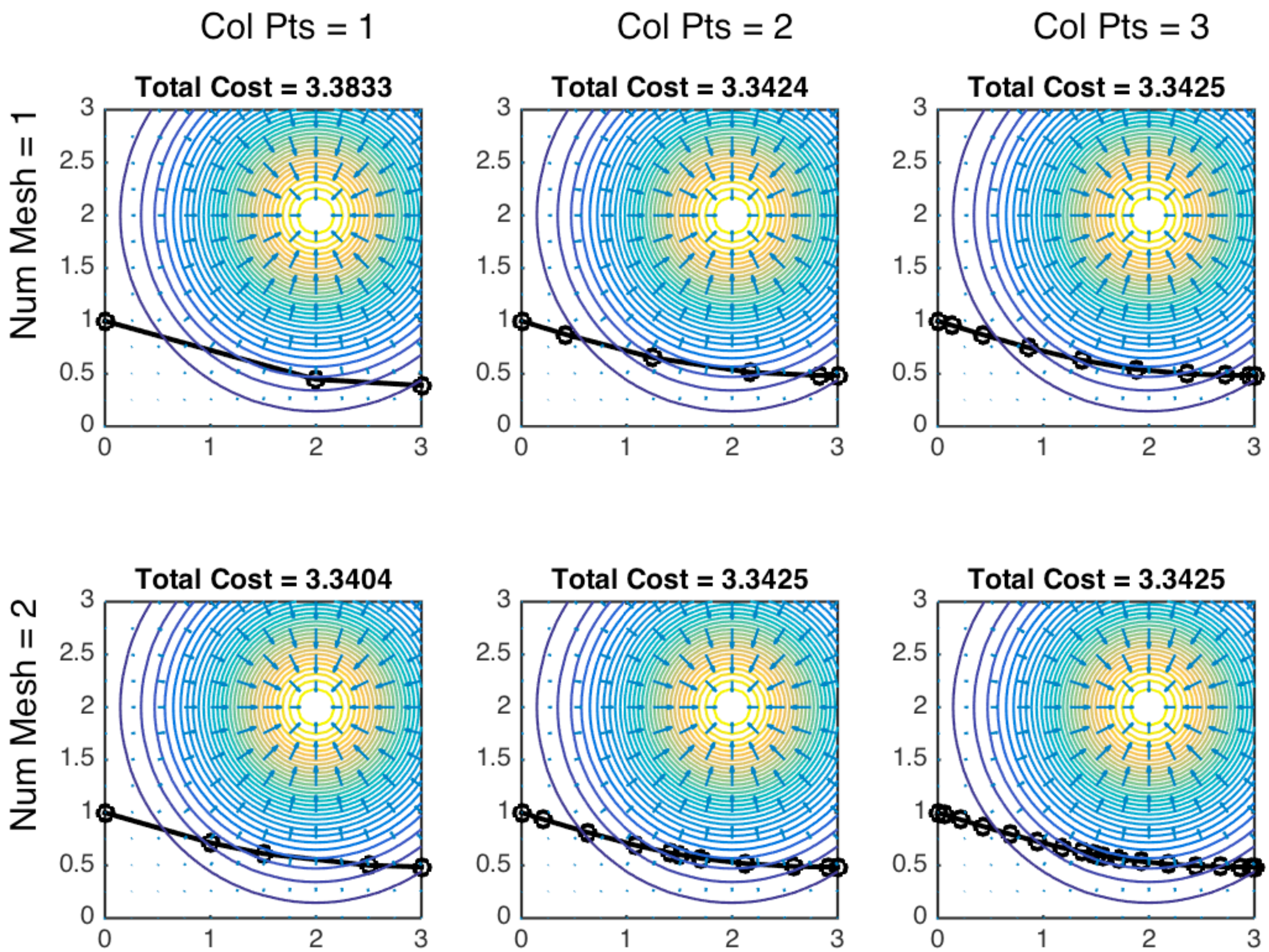
```
% compare the effects of using different numbers of collocation points and
% different numbers of mesh intervals.

clear all; close all; clc;

N_cols = [2 5 9];
N_mesh = [1 2];

figure('Name','Minimum Energy Path to the Road')
for i = 1:length(N_mesh)
    n_mesh = N_mesh(i);
    for j = 1:length(N_cols)
        n_cols = N_cols(j);
        Q4_Main;
        Q4_PlotComparison
    end
end
```

```
%---------------------------------------------%
% BEGIN: function Q4_Continuous.m %
%---------------------------------------------%
function phaseout = Q4_Continuous(input)
% the "output" from the continuous function includes "dynamics" (states)
% "path" (path constraints), and "integrand" (trajectory constraints).
% sprintf('Entering Continuous')

% global integrand

%% Dynamics
% store auxiliary data locally
% f = input.auxdata.f;

% store x-position locally
x  = input.phase.time;

% store states locally
y = input.phase.state(:,1);

% store controls locally
u = input.phase.control(:,1);

% dynamics equations
ydot = u;

phaseout.dynamics = [ydot];

%% Path Constraint

% phaseout.path = vnorm;

%% Integral Constraint, Error Minimization
% There is no integral constraint
% integrand for energy (power)

E = (1+u.^2).*(1+exp(-((x-2).^2+(y-2).^2)));
% E = (1+exp(-((x-2).^2+(y-2).^2)));

phaseout.integrand = E;

% sprintf('Leaving Continuous')
end
%---------------------------------------------%
% END: function Q4_Continuous.m %
%---------------------------------------------%
```

```
%------------------------------------------%
% BEGIN: function Q4_Endpoint.m            %
%------------------------------------------%
function output = Q4_Endpoint(input)
% The possible outputs are "objective" and "eventgroup" I don't know what
% "eventgroup" is, but I'm pretty sure we are not using it.
% sprintf('Entering Endpoint')

E = input.phase.integral; % This is the expended energy

output.objective = E;

% sprintf('Leaving Endpoint')
end
%------------------------------------------%
% END: function Q4_Endpoint.m              %
%------------------------------------------%
```

```
% Plot the comparisons

x = solution.phase.time;
y = solution.phase.state;
t_cost = solution.phase.integral;
M = length(N_mesh);
C = length(N_cols);
num = C*(i-1) + j;

% figure()
subplot(M,C,num)
plot(x,y,'k-o','linewidth',2)
hold on
Q4_GradientPlot
hold off
title(['Total Cost = ',num2str(t_cost)])
xlim([x0 xf]); ylim([ymin ymax]);
axis equal

if j == 1
    h1 = text(-0.75,0.8,['Num Mesh = ', num2str(i)]);
    set(h1,'rotation',90,'fontsize',14)
end

if i == 1
    h2 = text(0.9,3.75,['Num Col Pts = ', num2str(j)]);
    set(h2,'fontsize',14)
end

if num == M*C
    suptitle('')
end
```

```
% clear all; close all; clc;
syms x y
f = exp(-((x-2)^2+(y-2)^2));

gradf = jacobian(f,[x,y]);

[xx, yy] = meshgrid(x0:.1:xf,ymin:.1:ymax);
ffun = @(x,y) eval(vectorize(f));
fxfun = @(x,y) eval(vectorize(gradf(1)));
fyfun = @(x,y) eval(vectorize(gradf(2)));
contour(xx, yy, ffun(xx,yy), 30)
hold on
[xx, yy] = meshgrid(x0:.25:xf,ymin:.25:ymax);
quiver(xx, yy, fxfun(xx,yy), fyfun(xx,yy), 0.6)
axis equal tight, hold off
```


Col Pts = 1
Col Pts = 2
Col Pts = 3
Num Mesh = 1
Num Mesh = 2
Total Cost = 3.3833
Total Cost = 3.3424
Total Cost = 3.3425
Total Cost = 3.3404
Total Cost = 3.3425
Total Cost = 3.3425