CMPT 882 Assignment 2 Solutions

1. Sequential Quadratic Programming. Use Sequential Quadratic Programming to solve the NLP, introduced in class. You may use any software to solve the quadratic subproblems. (The solution will be given in cvx).

$$\begin{array}{ll} \text{minimize} & \sin(\pi x)\sin(2\pi y) \\ \text{subject to} & -1 \le x \le 1 \\ & -1 \le y \le 1 \end{array}$$

a) Construct a quadratic subproblem that minimizes the quadratic approximation of the objective, subject to linearization of the constraints, centred around a given iterate x_k.

Solution:

Since our constraints are already linear, in quadratic subproblem we only need to quadratize the objective. Thus, the quadratic subproblem is given by

$$\begin{array}{ll} \text{minimize} & \nabla f(x_k, y_k)^\top d_k + \frac{1}{2} d_k^\top Hf(x_k, y_k) d_k \\ \text{subject to} & -x - 1 \le 0 \\ & \;\; x - 1 \le 0 \\ & -y - 1 \le 0 \\ & \;\; y - 1 \le 0 \end{array}$$

Since $x = x_k + d_k$, we can rewrite the quadratic subproblem as follows:

$$\begin{array}{ll} \text{minimize} & \nabla f(x_k, y_k)^\top d_k + \frac{1}{2} d_k^\top Hf(x_k, y_k) d_k \\ \text{subject to} & -x_k - d_{k,x} - 1 \le 0 \\ & \;\; x_k + d_{k,x} - 1 \le 0 \\ & -y_k - d_{k,y} - 1 \le 0 \\ & \;\; y_k + d_{k,y} - 1 \le 0 \end{array}$$

where $d_k = \begin{bmatrix} d_{k,x} \\ d_{k,y} \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix}$

$$f(x, y) = \sin(\pi x)\sin(2\pi y)$$

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} \pi \cos(\pi x)\sin(2\pi y) \\ 2\pi \sin(\pi x)\cos(2\pi y) \end{bmatrix}$$

$$Hf(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} -\pi^2 \sin(\pi x)\sin(2\pi y) & 2\pi^2 \cos(\pi x)\cos(2\pi y) \\ 2\pi^2 \cos(\pi x)\cos(2\pi y) & -4\pi^2 \sin(\pi x)\sin(2\pi y) \end{bmatrix}$$

b) Write code to solve the above quadratic subproblem using cvx.

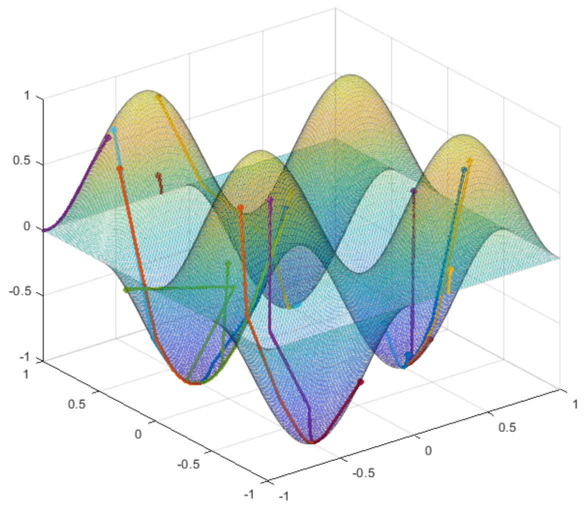
```
function dk = solve_quad_subproblem(xk, yk)
% Solves the quadratic subproblem using cvx
```

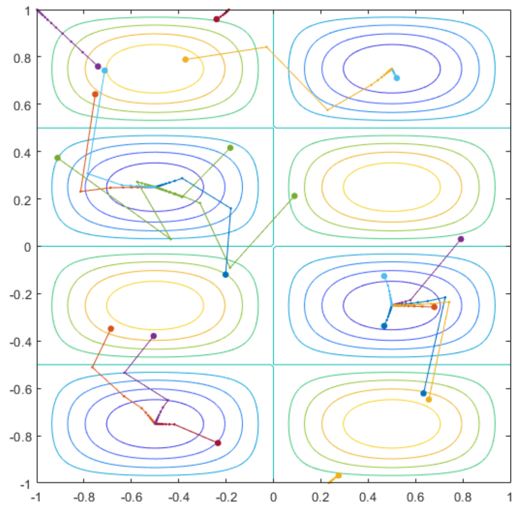
```
dz = grad_fun(xk, yk); % gradient
Hz = hess_fun(xk, yk); % Hessian

% Convexify Hessian
[V,D] = eig(Hz);
D(D<=0) = 0;
Hz = V*D*V^-1;

% Solve problem in cvx
cvx_quiet true
cvx_begin
  % Decision variable
  variable dk(2)

  % Quadratic objective
  minimize ( 0.5*quad_form(dk, Hz) + dz'*dk )

  % Constraints
  -xk - dk(1) - 1 <= 0
  xk + dk(1) - 1 <= 0
  -yk - dk(2) - 1 <= 0
  yk + dk(2) - 1 <= 0

cvx_end

end

function z = fun(x,y)
z = sin(pi*x).*sin(2*pi*y);
end


function dz = grad_fun(x,y)
% Gradient of objective function
dz = [pi*cos(pi*x)*sin(2*pi*y); 2*pi*sin(pi*x)*cos(2*pi*y)];
end

function Hz = hess_fun(x,y)
% Hessian of objective function
z11 = -pi^2 * sin(pi*x) * sin(2*pi*y);
z12 = 2*pi^2*cos(pi*x)*cos(2*pi*y);
z22 = -4*pi^2*sin(pi*x)*sin(2*pi*y);
Hz = [z11 z12; z12 z22];
end
```

c) Write code to solve the entire NLP, starting from several different initial points x_0. Make a plot showing several sequences of $\{x_k\}$ starting these initial points.

```
% Grid for plotting
X = -1:0.01:1;
Y = -1:0.01:1;
[x,y] = ndgrid(X,Y);

f = figure;
```

```
f.Position = [100 100 1600 600];
subplot(1,2,1) % Surf (3D) plot
surf(x,y,fun(x,y), "edgealpha", 0.1, "facealpha", 0.5)
drawnow
hold on

subplot(1,2,2) % Contour (2D) plot
contour(x,y,fun(x,y))
axis square
drawnow
hold on

num_ics = 20; % Number of initla conditions
colors = lines(num_ics);

maxIter = 25;
small = 1e-3; % Tolerance for convergence
alpha = 0.25; % Step size

for ii = 1:num_ics
  % Random initial conditions
  xk = -1+2*rand;
  yk = -1+2*rand;

  % Sequence of iterates
  xks = xk;
  yks = yk;

  % Plot initial conditions
  subplot(1,2,1)
  plot3(xk, yk, fun(xk,yk), '.', 'color', colors(ii,:), 'markersize', 15);
  drawnow;

  subplot(1,2,2)
  plot(xk, yk, '.', 'color', colors(ii,:), 'markersize', 15);
  drawnow;


  % Repeatedly solve quadratic subproblem
  for i = 1:maxIter

    dk = solve_quad_subproblem(xk, yk);
    xk = xk + alpha*dk(1);
    yk = yk + alpha*dk(2);
    xks = [xks xk];
    yks = [yks yk];

    % Convergence condition
    if norm(dk) <= small
      break
    end
  end

  % Plot sequence of iterates
  subplot(1,2,1)
  plot3(xks, yks, fun(xks,yks), '.-', 'color', colors(ii,:), ...
    "linewidth", 2.5);
  drawnow;

  subplot(1,2,2)
  plot(xks, yks, '.-', 'color', colors(ii,:));
  drawnow;
end
```





2. Differential Flatness. Consider the following simple model of the car (that is slightly more complex than the one introduced in class:

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \\ \dot{v} &= a\end{aligned}$$

The states consist of the position (x, y), the heading θ, and the longitudinal speed v. The controls are the turn rate ω and the longitudinal acceleration a.

a) Show that the system is differentially flat by letting $z = (x, y)$, and deriving the functions β and γ such that $(x, y, \theta, v) = \beta\left(z, \dot{z}, \dots, z^{(q)}\right), (\omega, a) = \gamma\left(z, \dot{z}, \dots, z^{(q)}\right)$.

Solution:

For the function β, we need to write each of the state variables as a function of z and its derivatives only, or x and y and their derivatives only.

First, similar to our discussion in class, we have

$$\begin{aligned}\frac{\dot{x}}{\dot{y}} &= \tan\theta \\ \theta &= \arctan\left(\frac{\dot{y}}{\dot{x}}\right)\end{aligned}$$

Next, also similar to our discussion in class, we have

$$v = \frac{\dot{x}}{\cos\theta}$$

(Alternatively, $v = \sqrt{(\dot{x})^2 + (\dot{y})^2}$ works too.)

The difference now is that v is a state instead of a constant. Here, note that θ can be written as a function of $\dot{x}$ and $\dot{y}$, so v can also be written as a function of derivatives of x and y – namely $\dot{x}$ and $\dot{y}$. Summarizing, we have

$$\begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \\ \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \\ \dfrac{\dot{x}}{\cos\theta} \end{bmatrix}$$

Again, note that the entire right-hand side can be written as a function of $x, y, \dot{x}, \dot{y}$.

To obtain function γ, we can simply take the derivative of θ and v, respectively. Since θ and v can be written in terms of $x, y, \dot{x}, \dot{y}$, so too can $\dot{\theta}$ and $\dot{v}$. Thus, without writing out ω and a explicitly in terms of x, y and their derivatives, we simply have $\begin{bmatrix} \omega \\ a \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \dot{v} \end{bmatrix}$, where θ and v can be written in terms of x, y and their derivatives.

b) Consider a maneuver which reverses the direction of the car, given by the following initial condition at $t = 0$ and final condition at $t = T = 10$:

$$\begin{bmatrix} x(0) \\ y(0) \\ \theta(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} x(T) \\ y(T) \\ \theta(T) \\ v(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{\pi}{2} \\ 1 \end{bmatrix}$$

Using the basis functions $\psi_0(t) = 1, \psi_1(t) = t, \psi_2 = t^2, \psi_3 = t^3, \psi_4 = t^4$, consider the following parametrization of the flat outputs

$$x(t) = \sum_{i=0}^{3} b_{0i}\psi_i(t)$$

$$y(t) = \sum_{i=0}^{3} b_{1i}\psi_i(t)$$

Write a system of equations in terms of the coefficients $\{b_{0i}\}$ and $\{b_{1i}\}$ such that the initial and final conditions are satisfied.

For clarity, we first explicitly write the parametrizations of $x(t)$ and $y(t)$:

$$x(t) = b_{00} + b_{01}t + b_{02}t^2 + b_{03}t^3$$
$$y(t) = b_{10} + b_{11}t + b_{12}t^2 + b_{13}t^3$$

Taking their derivatives, we get

$$\dot{x}(t) = b_{01} + 2b_{02}t + 3b_{03}t^2$$
$$\dot{y}(t) = b_{11} + 2b_{12}t + 3b_{13}t^2$$

The initial and final conditions are given in the original state space, and we need to first write them in the flat output space involving only x, y and their derivatives. The conditions on x and y are straight forward, since they are the flat outputs. We can obtain the conditions on $\dot{x}$ and $\dot{y}$ from the dynamics:

$$\dot{x}(0) = v(0)\cos(\theta(0)) = 1$$
$$\dot{x}(T) = v(T)\cos(\theta(T)) = 0$$
$$\dot{y}(0) = v(0)\sin(\theta(0)) = 0$$
$$\dot{y}(T) = v(T)\sin(\theta(T)) = 1$$

Thus, to summarize, the constraints representing initial and final conditions in the flat output space are as follows:

$$\begin{bmatrix} x(0) \\ \dot{x}(0) \\ x(T) \\ \dot{x}(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} y(0) \\ \dot{y}(0) \\ y(T) \\ \dot{y}(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Incorporating the parametrizations of $x, \dot{x}, y, \dot{y}$, we have

For x and $\dot{x}$, $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} b_{00} \\ b_{01} \\ b_{02} \\ b_{03} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

For y and $\dot{y}$, $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} b_{10} \\ b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

c) Solve the above systems of equations using a software of your choice to obtain and plot the state and control trajectories.

To obtain the state and control trajectories from the flat outputs and their derivatives we use the result from part a):

$$\begin{bmatrix} x \\ y \\ \theta \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \\ \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \\ \frac{\dot{x}}{\cos\theta} \end{bmatrix}$$

One caveat here is division by zero. For $\arctan\left(\frac{\dot{y}}{\dot{x}}\right)$, a proper arctangent function (`atan2` in Matlab) would handle division by zero. For $\frac{\dot{x}}{\cos\theta}$, note that whenever $\cos\theta = 0$, we must have $\theta = \pm\frac{\pi}{2}$, which means $\dot{x} = 0$, since the vehicle is traveling up or down. In this case, we can use the equation $\dot{y} = v\sin\theta$ to compute v: $v = \frac{\dot{y}}{\sin\theta}$. Here, $\sin\theta = \pm 1 \neq 0$ whenever $\cos\theta = 0$.

Note that another way to avoid this division by zero is to write $v = \sqrt{(\dot{x})^2 + (\dot{y})^2}$. We will use this option later, just to show both ways.

For the controls, we start from

$$\begin{bmatrix} \omega \\ a \end{bmatrix} = \begin{bmatrix} \frac{d}{dt}\left(\arctan\left(\frac{\dot{y}}{\dot{x}}\right)\right) \\ \dot{v} \end{bmatrix}$$

and take derivatives, since $\omega = \dot{\theta}$:

$$\begin{aligned} \omega(t) &= \frac{d}{dt}\left(\arctan\left(\frac{\dot{y}}{\dot{x}}\right)\right) \\ &= \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{(\dot{x})^2\left(1 + \left(\frac{\dot{y}}{\dot{x}}\right)^2\right)} \\ &= \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{(\dot{x})^2 + (\dot{y})^2} \\ &= \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{v^2} \end{aligned}$$

where $\ddot{x} = 2b_{02} + 6b_{03}t$, and $\ddot{y} = 2b_{12} + 6b_{13}t$.

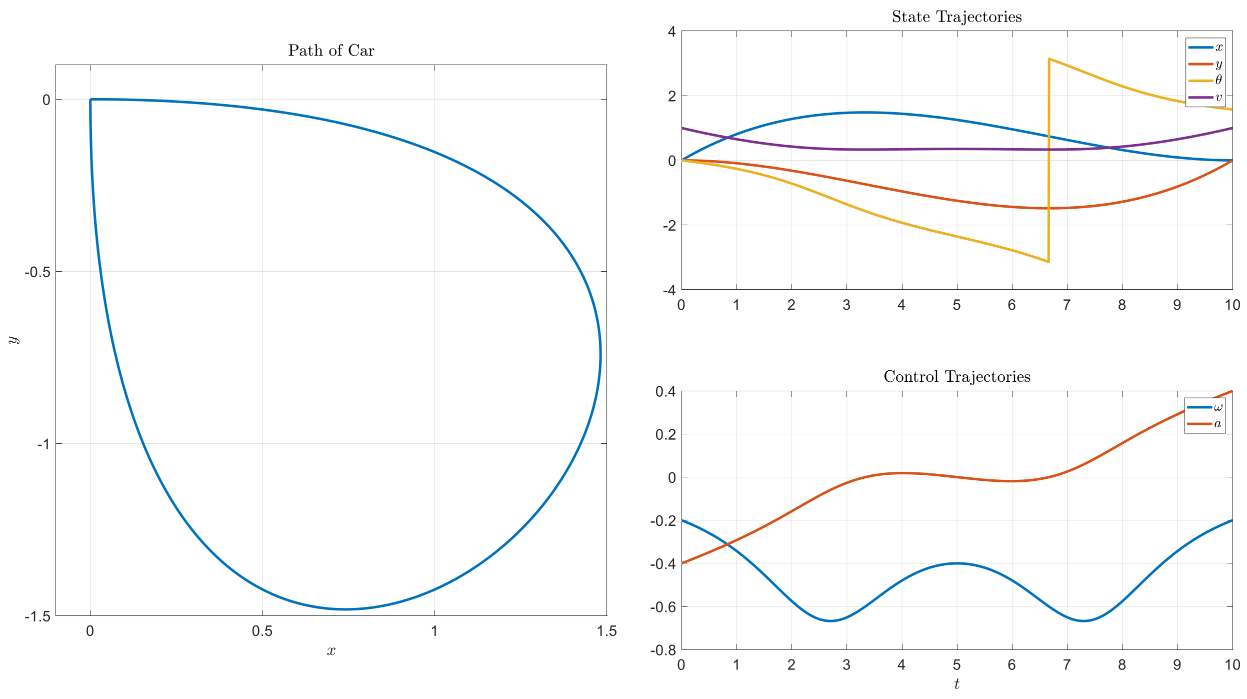
Lastly, for v, this time we will use the relation $v^2 = (\dot{x})^2 + (\dot{y})^2$ to avoid division by zero. We will differentiate this equation implicitly, and isolate $\dot{v}$, since $a = \dot{v}$:

$$\begin{aligned} v^2 &= (\dot{x})^2 + (\dot{y})^2 \\ 2v\dot{v} &= 2\dot{x}\ddot{x} + 2\dot{y}\ddot{y} \\ va(t) &= \dot{x}\ddot{x} + \dot{y}\ddot{y} \\ a(t) &= \frac{\dot{x}\ddot{x} + \dot{y}\ddot{y}}{v} \end{aligned}$$

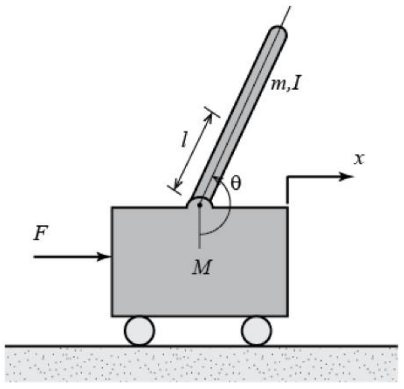
Code:

```
%% Computing flat outputs
% Time horizon
T = 10;

% Matrix for linear equation representing constraints in flat output space
M = [1 0 0 0; 0 1 0 0; 1 T T^2 T^3; 0 1 2*T 3*T^2];

% RHS of constraints in flat output space
x_constr = [0; 1; 0; 0];
y_constr = [0; 0; 0; 1];

% Solve for coefficients of basis functions
b0 = M\x_constr;
b1 = M\y_constr;
% Note: lengths of b0 and b1: 4

%% Computing state and control trajectories from flat outputs
dt = 0.01;
t = 0:dt:T;

% First, we compute x, y, and their derivatives; these are analytic since
% the basis functions are monomials
x = zeros(size(t));
y = zeros(size(t));
% x and y have powers of t from 0 to length(b0)-1
for i = 0:length(b0)-1
    x = x + b0(i+1) * t.^i;
    y = y + b1(i+1) * t.^i;
end

xdot = zeros(size(t));
ydot = zeros(size(t));
% xdot and ydot have powers of t from 0 to length(b0)-2
for i = 0:length(b0)-2
    xdot = xdot + (i+1)*b0(i+2)*t.^i;
    ydot = ydot + (i+1)*b1(i+2)*t.^i;
end

xddot = zeros(size(t));
yddot = zeros(size(t));
% xddot and yddot have powers of t from 0 to length(b0)-3
```

```matlab
for i = 0:length(b0)-3
    xddot = xddot + factorial(i+2)*b0(i+3)*t.^i;
    yddot = yddot + factorial(i+2)*b1(i+3)*t.^i;
end

% Finally, we have the theta and v trajectories (x and y trajectories have
% already been computed)
theta = atan2(ydot, xdot);
v = xdot./cos(theta);
% Use alternate expression for v when cos(theta) == 0
% Other option: use v = sqrt(xdot^2 + ydot^2)
small = 1e-5;
cos_zero_inds = cos(theta) < small;
v(cos_zero_inds) = ydot(cos_zero_inds)./sin(theta(cos_zero_inds));

% Control trajectories
omega = (yddot.*xdot - xddot.*ydot)./v.^2;
a = (xdot.*xddot + ydot.*yddot)./v;

%% Plot
f1 = figure;
f1.Position = [100 100 2400 1200];

% Path in x-y space
s1 = subplot(2,4,[1 2 5 6]);
plot(x,y, "linewidth", 3);
title("Path of Car", "Interpreter", "Latex")
xlabel("$x$", "Interpreter", "Latex")
ylabel("$y$", "Interpreter", "Latex")
grid on
s1.FontSize = 18;

% State trajectory over time
s2 = subplot(2,4,[3 4]);
plot(t,x, "linewidth", 3);
hold on
plot(t,y, "linewidth", 3);
plot(t,theta, "linewidth", 3);
plot(t,v, "linewidth", 3);
title("State Trajectories", "Interpreter", "Latex")
legend({"$x$", "$y$", "$\theta$", "$v$"}, "Interpreter", "Latex")
grid on
s2.FontSize = 18;

% Control trajectory over time
s3 = subplot(2,4,[7 8]);
plot(t,omega, "linewidth", 3)
hold on
plot(t,a, "linewidth", 3)
title("Control Trajectories", "Interpreter", "Latex")
legend({"$\omega$", "$a$"}, "Interpreter", "Latex")
xlabel("$t$", "Interpreter", "Latex")
grid on
s3.FontSize = 18;
```


Path of Car
0
-0.5
-1
-1.5
0
0.5
1
1.5
x
y
State Trajectories
x
y
θ
v
4
2
0
-2
-4
0 1 2 3 4 5 6 7 8 9 10
Control Trajectories
ω
a
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0 1 2 3 4 5 6 7 8 9 10
t

3. Multiple Shooting with Casadi.



Consider a system involving an inverted pendulum on a cart. For this system, the state is (x, v, θ, ω), representing the position of the cart, speed of the cart, angle of the pendulum, and angular speed of the pendulum, respectively. The initial state at time $t = 0$ is $(0,0,0,0)$.

The problem parameters are as follows: $M = 0.5$ kg is the mass of the cart, $m = 0.2$ kg is the mass of the pendulum, $l = 0.3$ m is the half-length of the pendulum, $I = 0.006$ kg·m^2 is the moment of inertial of the pendulum.

We would like to apply a force $F(t)$ such that at time $t = 5$ s, the cart is stopped at the origin, with the pendulum being stationary at the upright position, represented by the state $(0,0,\pi,0)$. At the same time, we would like to minimize the control effort, given by the integral $\int_{t=0}^{5} F^2(t)dt$, while ensuring that the position of the cart stays within 1 m of the origin, $|x| \leq 1$. The control is also limited by $|F(t)| \leq 0.2$ at all times.

The equations of motion are given by

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F$$
$$(I+ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta$$

where $b = 0.1$ N/m/s is the coefficient of friction, and $g = 9.81$ m/s/s is the acceleration due to gravity on Earth. In terms of a first-order ODE, we have $\dot{x} = v, \dot{\theta} = \omega$. The rest of the system dynamics, for (v, ω), can be implicitly written as

$$\begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I+ml^2 \end{bmatrix}\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -bv + ml\omega^2\sin\theta + F \\ -mgl\sin\theta \end{bmatrix}$$

a) Write down the associated optimal control problem.

Solution:

$$\begin{aligned}
\underset{u(\cdot)}{\text{minimize}} \quad & \int_{t=0}^{5} F^2(t)\,dt \\
\text{subject to} \quad & x(0) = 0 \\
& v(0) = 0 \\
& \theta(0) = 0 \\
& \omega(0) = 0 \\
& x(5) = 0 \\
& v(5) = 0 \\
& \theta(5) = \pi \\
& \omega(5) = 0 \\
& \dot{x}(t) = v(t) \\
& \dot{\theta}(t) = \omega(t)
\end{aligned}$$

$$\begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I+ml^2 \end{bmatrix}\begin{bmatrix} \dot{v}(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} -bv + ml\omega^2\sin\theta + F \\ -mgl\sin\theta \end{bmatrix}$$

$$|x(t)| \leq 1$$

$$|u(t)| \leq 0.2$$

b) Discretize the optimal control problem using the multiple shooting method and write down the resulting nonlinear program. Use $N = 50$ time intervals, forward Euler integration, and left endpoint first order integration (the same schemes as presented in the lecture).

Let $N = 50, h = \frac{T}{N}$, and we get the following nonlinear optimization program:

$$\begin{aligned}
\underset{\{(x_k,v_k,\theta_k,\omega_k)\}_{k=0}^{N},\{F_k\}_{k=0}^{N-1}}{\text{minimize}} \quad & h\sum_{k=0}^{N-1} F_k^2 \\
\text{subject to} \quad & x_0 = 0 \\
& v_0 = 0 \\
& \theta_0 = 0 \\
& \omega_0 = 0 \\
& x_N = 0 \\
& v_N = 0 \\
& \theta_N = \pi \\
& \omega_N = 0 \\
& \forall k \in \{0,1,\dots,N\}, \\
& \quad x_{k+1} = x_k + hv_k \\
& \quad v_{k+1} = v_k + h\dot{v}_k \\
& \quad \theta_{k+1} = \theta_k + h\omega_k \\
& \quad \omega_{k+1} = \omega_k + h\dot{\omega}_k \\
& \quad \begin{bmatrix} M+m & ml\cos\theta \\ ml\cos\theta & I+ml^2 \end{bmatrix}\begin{bmatrix} \dot{v}_k \\ \dot{\omega}_k \end{bmatrix} = \begin{bmatrix} -bv + ml\omega^2\sin\theta + F \\ -mgl\sin\theta \end{bmatrix} \\
& \quad |x_k| \leq 1 \\
& \forall k \in \{0,1,\dots,N-1\}, \\
& \quad |u_k| \leq 0.2
\end{aligned}$$

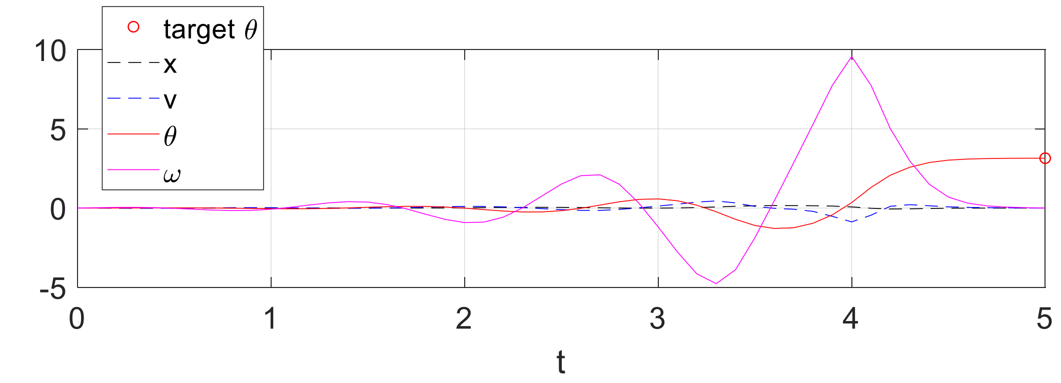
c) Using the Casadi toolbox, which can be found at https://github.com/casadi/casadi/wiki, solve the nonlinear program, and plot $(x(t), v(t), \theta(t), \omega(t))$, and $F(t)$. A good starting point is the direct multiple shooting example at https://github.com/casadi/casadi/wiki/examples.

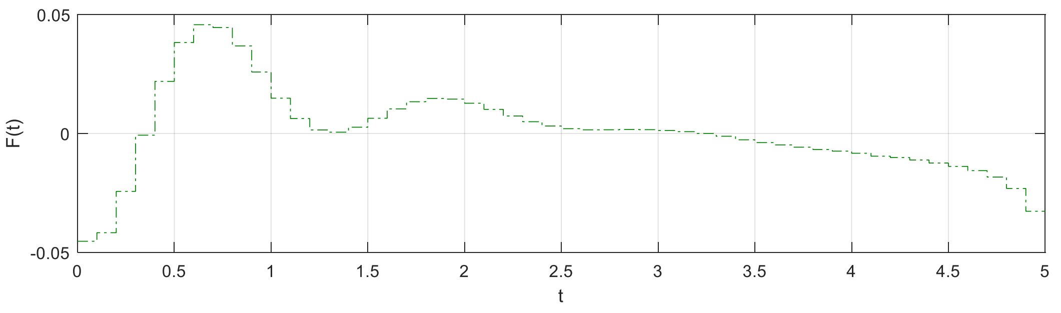
```
% This file has been modified from the multiple shooting example
% in the Casadi toolbox

import casadi.*

T = 5; % Time horizon
N = 50; % number of control intervals

%% Declare model variables
x = SX.sym('x');
```

```matlab
v = SX.sym('v');
theta = SX.sym('theta');
omega = SX.sym('omega');

state = [x; v; theta; omega];
u = SX.sym('u');

% Problem parameters
M = 0.5;
m = 0.2;
b = 0.1;
l = 0.3;
I = 0.006;
g = 9.81;
uMax = 0.2;

%% Dynamics and cost
% Write \dot v and \dot\omega as the solution of a linear system of
% equations
xdot24 = [M+m             m*l*cos(theta);
          m*l*cos(theta) I+m*l^2]        \ ...
         [-b*v + m*l*omega^2*sin(theta) + u;
          -m*g*l*sin(theta)];

xdot = [v; xdot24(1); omega; xdot24(2)];

% Running cost
Jt = u^2;

%% Forward Euler integration
dt = T/N;

f = Function('f', {state, u}, {xdot, Jt}); % f: two inputs two outputs

X0 = MX.sym('X0', 4);
U = MX.sym('U');

[Xdot, Jk] = f(X0, U);
X = X0 + dt*Xdot;
Q = Jk*dt;

% F has two inputs and two outputs, with the corresponding names
F = Function('F', {X0, U}, {X, Q}, {'x0','p'}, {'xf', 'qf'});

% Evaluate at a test point
Fk = F('x0',[0; 0; pi; 0],'p',1);
disp(Fk.xf)
disp(Fk.qf)

%% Construct the NLP
% Start with an empty NLP
```

```matlab
w={};
w0 = [];
lbw = [];
ubw = [];
J = 0;
g={};
lbg = [];
ubg = [];

% Constraints for initial state
Xk = MX.sym('X0', 4);
w = {w{:}, Xk};
lbw = [lbw; 0; 0; 0; 0];
ubw = [ubw; 0; 0; 0; 0];
w0 = [w0; 0; 0; 0; 0];

% Formulate the NLP
for k=0:N-1
  % New NLP variable for the control
  Uk = MX.sym(['U_' num2str(k)]);
  w = {w{:}, Uk};
  lbw = [lbw; -uMax];  % Control bounds
  ubw = [ubw;  uMax];
  w0 = [w0;  0];       % Initial guess for control

  % Integrate dynamics and cost
  Fk = F('x0', Xk, 'p', Uk);
  Xk_end = Fk.xf;
  J=J+Fk.qf;

  % New NLP variable for state at end of interval
  Xk = MX.sym(['X_' num2str(k+1)], 4);
  w = [w, {Xk}];

  if k == N-1
    % Constraints on state for final condition
    lbw = [lbw; 0; 0; pi; 0];
    ubw = [ubw;  0; 0; pi; 0];
  else
    % Constraints on state
    lbw = [lbw; -1; -inf; -inf; -inf];
    ubw = [ubw;  1;  inf; inf; inf];
  end

  w0 = [w0; 0; 0; 0; 0]; % Initial guess for state

  % Dynamics constraint
  g = [g, {Xk_end-Xk}];
  lbg = [lbg; 0; 0; 0; 0];
  ubg = [ubg; 0; 0; 0; 0];
```

```matlab
end

%% Create an NLP solver
prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob);

% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,...
  'lbg', lbg, 'ubg', ubg);
w_opt = full(sol.x);

%% Plot the solution
x_opt = w_opt(1:5:end);
v_opt = w_opt(2:5:end);
theta_opt = w_opt(3:5:end);
omega_opt = w_opt(4:5:end);
u_opt = w_opt(5:5:end);
tgrid = linspace(0, T, N+1);


f = figure;
f.Position = [100 100 960 600];
s1 = subplot(2,1,1);

htt = plot(tgrid(end), pi, 'ro');
hold on

hx = plot(tgrid, x_opt, 'k--');
hv = plot(tgrid, v_opt, 'b--');
ht = plot(tgrid, theta_opt, 'r-');
hw = plot(tgrid, omega_opt, 'm-');

xlabel('t')
legend([htt hx hv ht hw], {"target \theta", "x", "v", "\theta",
"\omega"})
grid on
s1.FontSize = 18;

s2 = subplot(2,1,2);
stairs(tgrid, [u_opt; nan], '-.','color',[0 0.5 0])
grid on
s1.FontSize = 18;
xlabel('t')
ylabel('F(t)')
```


target θ
x
v
θ
ω
10
5
0
-5
0
1
2
3
4
5
t


0.05
0
-0.05
F(t)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
t

4. Reachability Analysis. Consider a kinematic model of a fixed-wing aircraft flying in the presence of wind, given as follows:

$$\begin{aligned}\dot{x} &= v\cos\theta + d_x\\ \dot{y} &= v\sin\theta + d_y\\ \dot{\theta} &= u + d_\theta\end{aligned}$$

The state (x, y, θ) consists of x- and y- position, and heading θ. The aircraft flies at a constant speed $v = 0.3$ km/s, and controls its turn rate u, with $|u| \le 0.5$ rad/s. The effect of wind is modeled by the disturbance $d = (d_x, d_y, d_\theta)$ with $|d_x|, |d_y| \le 50$ m/s, and $|d_\theta| \le 0.005$ rad/s.

The initial position and heading of the aircraft is $(x, y, \theta) = (0,0,0)$, and the aircraft is flying towards its next waypoint at $x = 2$ km, $y = 2$ km, $\theta = \frac{3\pi}{8}$, with an acceptable tolerance of 200 m in distance and $\frac{\pi}{10}$ in heading. The aircraft needs to reach the waypoint within tolerance at time $t = 0$.

a) What is the target set $\mathcal{T}$ representing the acceptable set of states for reaching the waypoint within the tolerances? What is a suitable function $l(x, y, \theta)$ such that $l(x, y, \theta) \le 0 \Leftrightarrow (x, y, \theta) \in \mathcal{T}$?

Solution:

A linear expression in the value function is useful for defining half-spaces. For example, $\{(x, y, \theta): x \le 2.2\} = \{(x, y, \theta): x - 2.2 \le 0\}$. The max function is useful for taking the intersection of different sets. Combining the above two observations we have the following suitable function l:

$$l(x, y, \theta) = \max\left\{x - 2.2, \quad 1.8 - x, \quad y - 2.2, \quad 1.8 - y, \quad \theta - \frac{3\pi}{8} - \frac{\pi}{10}, \quad \frac{3\pi}{8} - \frac{\pi}{10} - \theta\right\}$$

Note that in the level set toolbox (https://bitbucket.org/ian_mitchell/toolboxls/overview), which the helperOC uses, we can use the shapeRectangleByCorners function to conveniently construct this set.

b) The value function $V(t, x, y, \theta)$ representing the set of states from which the aircraft can reach the target set within t seconds is the solution to the HJI PDE

$$\frac{\partial V}{\partial t}(t, x, y, \theta) + \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \nabla V(t, x, y, \theta)^\top f(x, y, \theta, u, d) = 0$$

where $f(x, y, \theta, u, d)$ is given by the system dynamics. Determine the optimal u and d optimizes the min-max expression. That is find an analytic expression for u^* and d^*, where

$$\begin{aligned}u^*(t, x, y, \theta) &= \arg\min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \nabla V(t, x, y, \theta)^\top f(x, y, \theta, u, d)\\ d^*(t, x, y, \theta) &= \arg\max_{d \in \mathcal{D}} \nabla V(t, x, y, \theta)^\top f(x, y, \theta, u^*, d)\end{aligned}$$

and $u \in \mathcal{U}$ and $d \in \mathcal{D}$ represents the bounds on the control and disturbances.

Solution:

As indicated by the system dynamics, $f(x, y, \theta, u, d)$ outputs a 3D vector, corresponding to $\dot{x}, \dot{y}, \dot{\theta}$. In particular,

$$f(x, y, \theta, u, d) = \begin{bmatrix} v\cos\theta + d_x \\ v\sin\theta + d_y \\ u + d_\theta \end{bmatrix}$$

The gradient of V is also a 3D vector, given by

$$\nabla V(t, x, y, \theta) = \begin{bmatrix} \frac{\partial V}{\partial x} \\ \frac{\partial V}{\partial y} \\ \frac{\partial V}{\partial \theta} \end{bmatrix}$$

We can now write down the dot product $\nabla V(t, x, y, \theta)^\top f(x, y, \theta, u, d)$:

$$\begin{aligned} \nabla V(t, x, y, \theta)^\top f(x, y, \theta, u, d) &= \frac{\partial V}{\partial x}(v\cos\theta + d_x) + \frac{\partial V}{\partial y}\left(v\sin\theta + d_y\right) + \frac{\partial V}{\partial \theta}(u + d_\theta) \\ &= \frac{\partial V}{\partial \theta}u + \frac{\partial V}{\partial x}d_x + \frac{\partial V}{\partial y}d_y + \frac{\partial V}{\partial \theta}d_\theta + \frac{\partial V}{\partial x}v\cos\theta + \frac{\partial V}{\partial y}v\sin\theta \end{aligned}$$

Note only the first term involves the control, the second to fourth terms only involve one component of disturbance, and the last two terms do not involve either the control or disturbance. This means that the control and each component of the disturbance can be optimized separately.

$$\begin{aligned} u^*(t, x, y, \theta) &= \arg\min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \nabla V(t, x, y, \theta)^\top f(x, y, \theta, u, d) \\ &= \arg\min_{|u| \le 0.5} \frac{\partial V}{\partial \theta} u \\ &= -0.5 \operatorname{sign}\left(\frac{\partial V}{\partial \theta}\right) \end{aligned}$$

where the $\operatorname{sign}(z) = 1$ if $z \ge 0$ and -1 otherwise.

Using a similar argument, but applied to maximization for d, we get

$$\begin{aligned} d_x &= 0.05 \operatorname{sign}\left(\frac{\partial V}{\partial x}\right) \\ d_y &= 0.05 \operatorname{sign}\left(\frac{\partial V}{\partial y}\right) \\ d_\theta &= 0.005 \operatorname{sign}\left(\frac{\partial V}{\partial \theta}\right) \end{aligned}$$

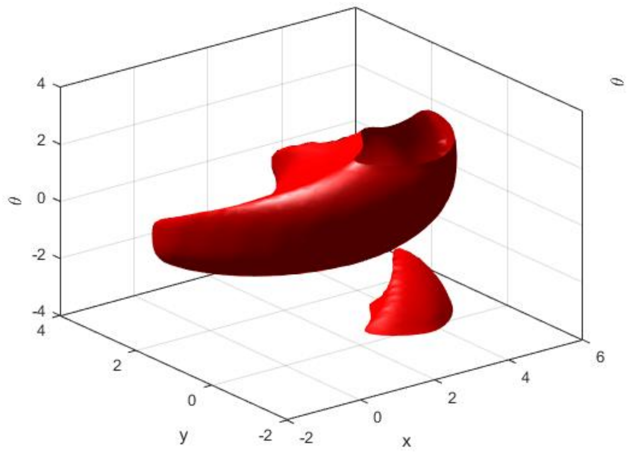
Note that all distances are measured in km.

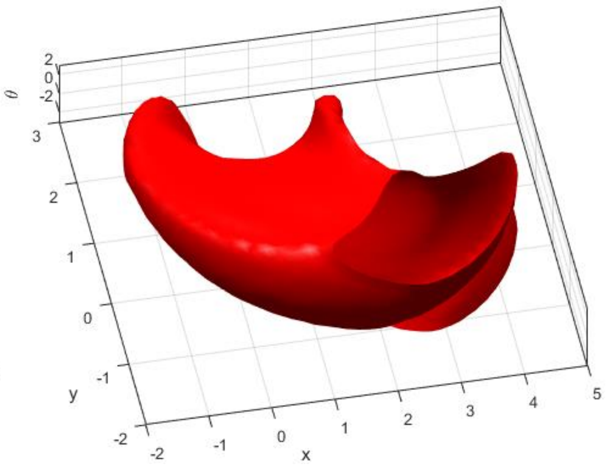
c) Compute $V(t,x,y,\theta)$ from $t=-10$ to $t=0$ using the helperOC toolbox, which can be found at https://github.com/HJReachability/helperOC. A good starting point is tutorial.m. Use the visSetIm and proj functions to visualize the following: $V(t=-10,x,y,\theta)$, $V(t=-5,x,y,\theta)$, $V(t=-10,x,y,\theta=0)$, $V(t=-10,x,y=0,\theta)$.

For your computation, use the following recommended grid bounds and resolution: $x \in [-2,5]$ with 45 grid points, $y \in [-2.5,4]$ with 45 grid points, $\theta \in [-\pi,\pi]$ with 35 grid points.

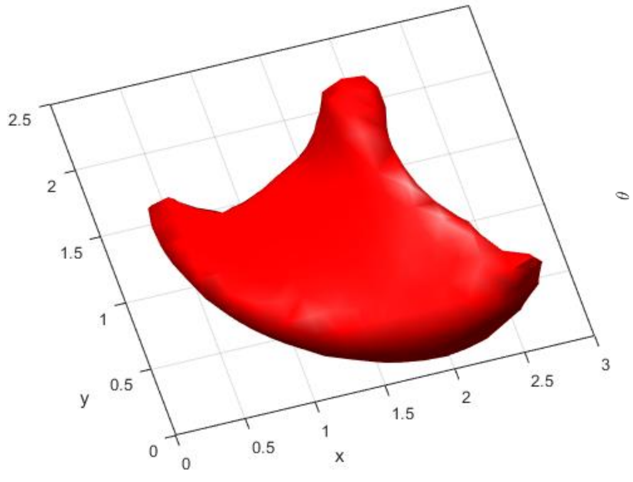
```
%% Grid
grid_min = [-2; -2.5; -pi]; % Lower corner of computation domain
grid_max = [5; 4; pi];    % Upper corner of computation domain
N = [45; 45; 35];         % Number of grid points per dimension
pdDims = 3;               % 3rd dimension is periodic
g = createGrid(grid_min, grid_max, N, pdDims);

%% target set
toler = [0.2; 0.2; pi/10];
goal = [2; 2; 3*pi/8];
lower = goal - toler;
upper = goal + toler;
data0 = shapeRectangleByCorners(g, lower, upper);

%% time vector
t0 = 0;
tMax = 10;
dt = 0.05;
tau = t0:dt:tMax;

%% problem parameters
speed = 0.4;
wMax = 0.5;
dMax = [0.05; 0.05; 0.005];

uMode = 'min';
dMode = 'max';

%% Pack problem parameters
% Define dynamic system
x0 = [0;0;0];
dCar = DubinsCar(x0, wMax, speed, dMax); %do dStep3 here

% Put grid and dynamic systems into schemeData
schemeData.grid = g;
schemeData.dynSys = dCar;
schemeData.accuracy = 'veryHigh'; %set accuracy
schemeData.uMode = uMode;
schemeData.dMode = dMode;

%% Compute value function
HJIextraArgs.visualize = true; %show plot
HJIextraArgs.fig_num = 1; %set figure number
HJIextraArgs.deleteLastPlot = true; %delete previous plot as you
update

data = HJIPDE_solve(data0, tau, schemeData, 'minVWithTarget',
HJIextraArgs);
save('a1q4.mat', 'tau', 'g', 'data')

%% Visualize slices
load('a1q4.mat')
% To visualize the final time, access the final 4th index
figure;
visSetIm(g, data(:,:,:,end));

% To visualize time slices, pick the correct 4th index
figure;
ind = find(tau==5);
visSetIm(g, data(:,:,:,ind));

% Now, we take other slices using the proj function
figure;
[gxy, dataxy] = proj(g, data(:,:,:,end), [0 0 1], 0);
visSetIm(gxy, dataxy);

figure;
[gxy, dataxy] = proj(g, data(:,:,:,end), [0 1 0], 0);
visSetIm(gxy, dataxy);
```

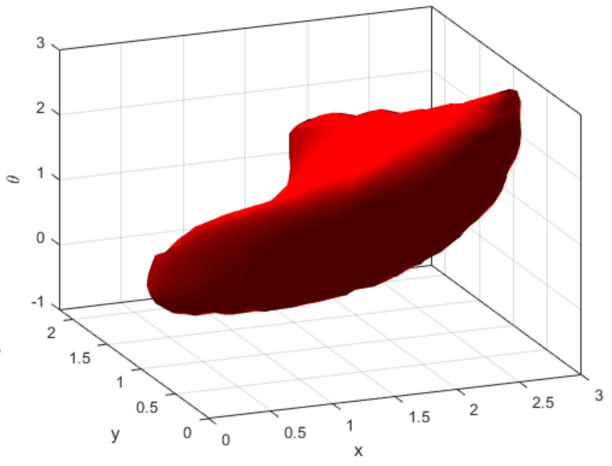
The zero sublevel set of $V(t = -10, x, y, \theta)$ is within the red surface. The extra piece at the bottom is a result of the periodicity of θ. The target set at the "tip" in the middle of the semicircular shape.



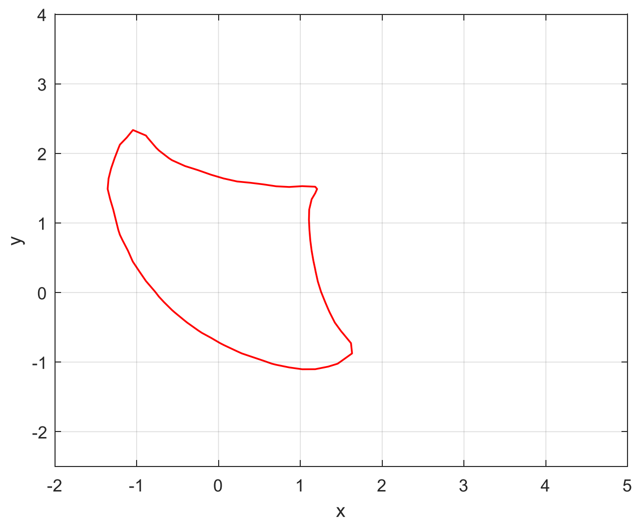


The zero sublevel set of $V(t=-5,x,y,\theta)$ is within the red surface.



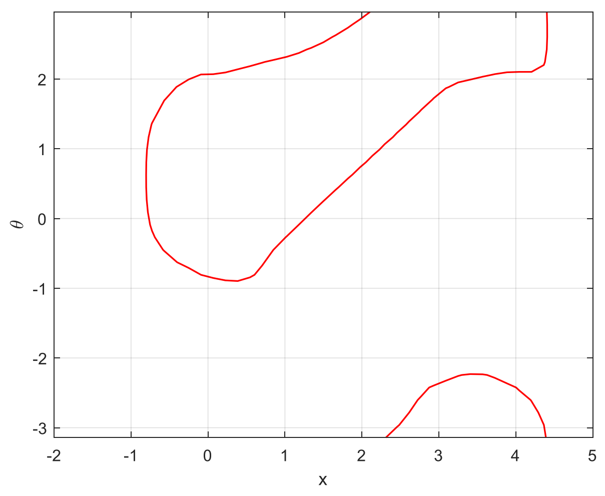


The zero sublevel set of $V(t=-10,x,y,\theta=0)$ is within the boundary below. The interpretation here is the set of positions from which the goal can be reached in 10 seconds, if the initial heading is 0 (facing the right side). Notably, this set does not contain the position of the target set, $(2{,}2)$, because even if the aircraft were at $(2{,}2)$, it would have the wrong heading.



The zero sublevel set of $V(t=-10,x,y=0,\theta)$ is within the boundary below. The interpretation here is the set of (x,θ) coordinates from which the goal can be reached in 10 seconds, if the initial y coordinate is 0. The two features that stand our are

1) The two "pieces" of sets caused by periodicity of θ
2) The "diagonal" shape, which indicates that the closer the initial heading is to 0 (facing right), the smaller the x coordinate values need to be. This makes sense, since the aircraft needs some room to turn to the goal region with the right heading.


θ
x
-2
-1
0
1
2
3
4
5
2
1
0
-1
-2
-3