6850

Project #3: tilt-a-whirl

MANE 4280/6710: Numerical Design Optimization

Fall 2021

Unique ID: 6850

Table of Contents

Executive Summary

Problem Statement

Design the structure of a Tilt-A-Whirl to maximize the standard deviation of the angular velocity of each car on the platform. (RPI, MANE4280/6710: Numerical Design Optimization, LMS, project-3.pdf)

Proposed Solution

The nominal values for the objective are $\omega = 6.5\times(2\pi/60)$ rad/s, r2 = 0.8 m, and α1 = 0.058 rad. They are based on the values in the paper. The objective values are in the range [1.49. 1.61]. The goal is to get a result 200% larger than the nominal value, then the value we are aiming for should be larger than 1.49*3 = 4.47. A reasonable integration period(T) needs to be found so the objective value will be in the range [1.49. 1.61]. The motion of the Tilt-A-Whirl using the equation in the paper, "*Chaos* *at the amusement park: Dynamics of the Tilt-A-Whirl*", R.L. Kautz and B.M. Huggard, Am. J. Phys. 62(1), January 1994.

$$\frac{d^2\phi}{d\tau^2}+(\gamma/Q_0)\frac{d\phi}{d\tau}+(\epsilon-\gamma^2\alpha)\sin\phi+\gamma^2\beta\cos\phi=0$$

The solution of this equation will be sent to the objective equation, the standard deviation of $\frac{d\phi}{dt}$ $over\ time$:

$$f(x)=3\omega\sqrt{\frac{1}{T}\int_0^T(y_1-\bar{y}_1)^2\ d\tau}$$

MATLAB function trapz (Trapezoidal numerical integration) will be used to calculate this equation. The objective function will be defined as go surrogate using an anonymous function. The optimal result can be calculated using MATLAB fmincon function (Find minimum of constrained nonlinear multivariable function) under correct constraints.

Results

The integration period, T, is set to 5. Number of samples used in the model are 200. Design variables are 3: angular velocity (ω), the radial arm from the center of each car platform to the center of the gravity of the car (r_2), and maximum/minimum incline angle (in radians) of the beam from the center of the ride to the car platform. Using $\omega = 0.08378\ rpm, r_2 = 0.2748\ m, and\ \alpha_1 = 0.03000\ rad$, we get the optimum objective value of 7.3059. It is 390.33% larger than the nominal value (1.49). All variables are under constraints.

Analysis Method

First, we need to sample the 3D design space using [lhsdesign]. They will be scaled, so they can fit in the lower an upper bounds. Then they will be used to evaluate the objective function. The objective function is the standard deviation of $\frac{d\phi}{dt}$ $over\ time$:

$$f(x)=3\omega\sqrt{\frac{1}{T}\int_0^T(y_1-\bar{y}_1)^2\ d\tau}$$

where

- $\bar{y}_1\ is\ the\ mean\ angular\ velocity, \frac{1}{T}\int_0^T y_1\ d\tau$
- T is the total (non-dimensional period of simulation)

The surrogate model will take the result from the objective and then process the optimization. Optimization procedure is discussed in the Optimization section.

Geometry Parameterization

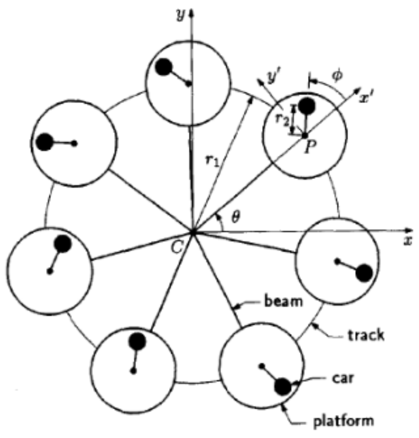


Figure 1: Tilt-A-Whirl Planview

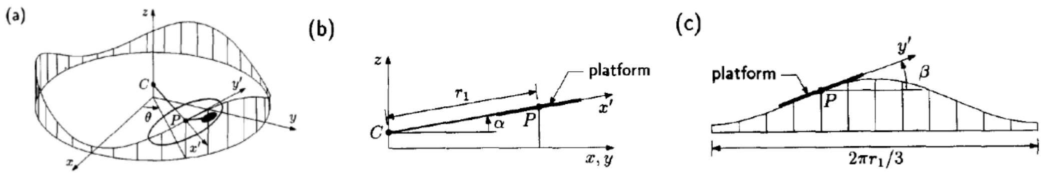


Figure 2 : Perspective view (a) showing a single platform, vertical cross section taken through points C and P (b), and vertical cross section taken through point P and tangent to the track.

The equation of motion is given in this paper, "*Chaos at the amusement park: Dynamics of the Tilt-A-Whirl*", R.L. Kautz and B.M. Huggard, Am. J. Phys. 62(1), January 1994.

$$\frac{d^2\phi}{d\tau^2}+(\gamma/Q_0)\frac{d\phi}{d\tau}+(\epsilon-\gamma^2\alpha)\sin\phi+\gamma^2\beta\cos\phi=0$$

Were

ϕ *is the car'sangle with respect to the beam*

Dimensionless parameters τ, γ, ϵ are:

$\tau = 3\omega t$, a nondimensional time

$$\gamma = \left(\frac{1}{3\omega}\right)\sqrt{\frac{g}{r2}}$$

$$\epsilon = \frac{r_1}{9r_2}$$

α *and* β in the equation are equal to:

$\alpha = \alpha_0 - \alpha_1 \cos(\tau)$

$\beta = 3\alpha_1 \sin(\tau)$

This equation is a second-order ordinary differential equation (ODE). The following steps shows how it is solved.

- $\frac{d^2\phi}{d\tau^2} = F[\phi, \frac{d\phi}{d\tau}] = -\left(\frac{\gamma}{Q_0}\right)\frac{d\phi}{d\tau} - (\epsilon - \gamma^2\alpha)sin\phi - \gamma^2\beta cos\phi$

- Define the state vector

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{d\phi}{d\tau} \\ \phi \end{bmatrix}$$

- add a new ODE to the system: $\frac{d\phi}{d\tau} = \frac{dy_2}{d\tau} = y_1$

so, we have

$$y_1 = \frac{d\phi}{d\tau}, y_2 = \phi$$

- Then the first-order system of ODEs to be solved is

$$\frac{dy}{d\tau} = \frac{d}{d_\tau}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} F(y_2, y_1) \\ y_1 \end{bmatrix}$$

$$y_1 = -\left(\frac{\gamma}{Q_0}\right) y_1 - (\epsilon - \gamma^2 \alpha) sin\, y_2 - \gamma^2 \beta cos\, y_2$$

$$y_2 = y_1$$

- Use MATLAB ODE45 to solve this system.
- In the code, we need to replace $\alpha\ and\ \beta$ use the equation below:

$$\alpha = \alpha_0 - \alpha_1 \cos(\tau)$$
$$\beta = 3\alpha_1 \sin(\tau)$$

After the equation is solved, the solution is sent to the objective function.

Optimization

Optimization statements

Find the max standard deviation of the angular velocity of each car on the platform.

Optimization methods

The solution of the motion equation of the Tilt-A Whirl is sent to the objective function. Then, surrogate model is used to approximate the objective function, and apply a gradient-based optimization to the surrogate. Design parameters and bound constrains are listed in the following section.

The results of the GP surrogate is set to f = -f, so fmincon can take this value to find the local minimum that satisfies all the constraints.

Design Variables

- ω: the angular velocity of overall ride, $\omega = \theta^{\cdot}$
- r_2: the radial arm from the center of each car platform to the center of gravity of the car
- α_1: the maximum/minimum incline angle (in radians) of the beam from the center of the ride to the car platform.

Constraints

- $3\, rpm \leq \omega \frac{60}{2\pi} rpm \leq 8\, rpm$
- $0.1\, m \leq r_2 \leq 1.5\, m$
- $0\, rad \leq \alpha_1 \leq 0.3\, rad$

Constants

- $r_1 = 4.3\, m$, the beam length from the center of the ride to the center of each platform

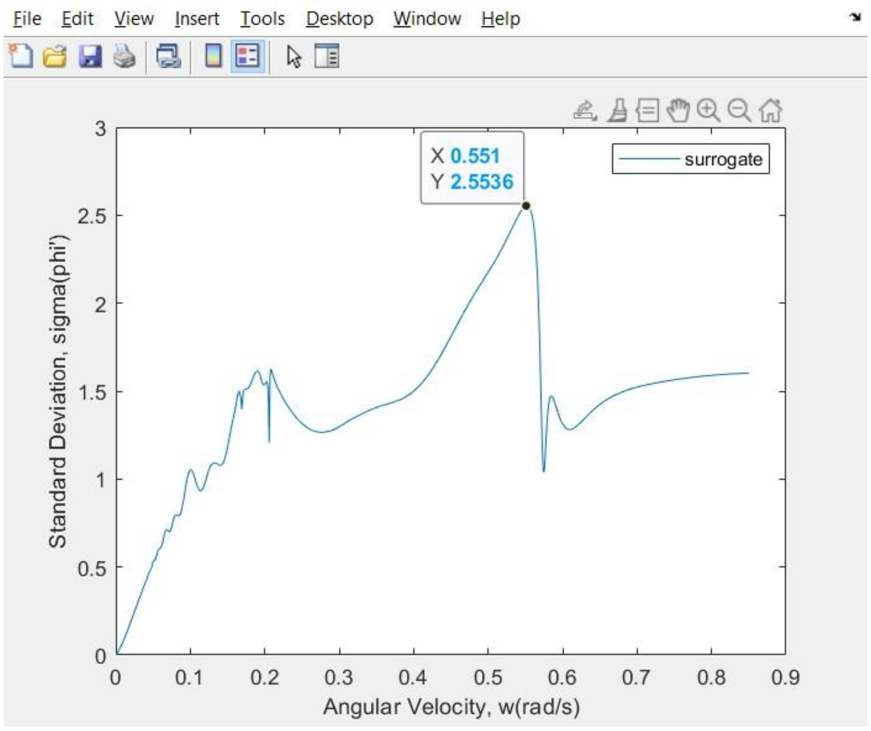
- $\alpha_0 = 0.036\,rad$, the incline offset
- $Q_0 = \left(\frac{m}{\rho}\right)\sqrt{g r_2^3} = 20$,
    - m : mass of the car
    - ρ : damping coefficient to account for friction
    - g : the acceleration due to gravity

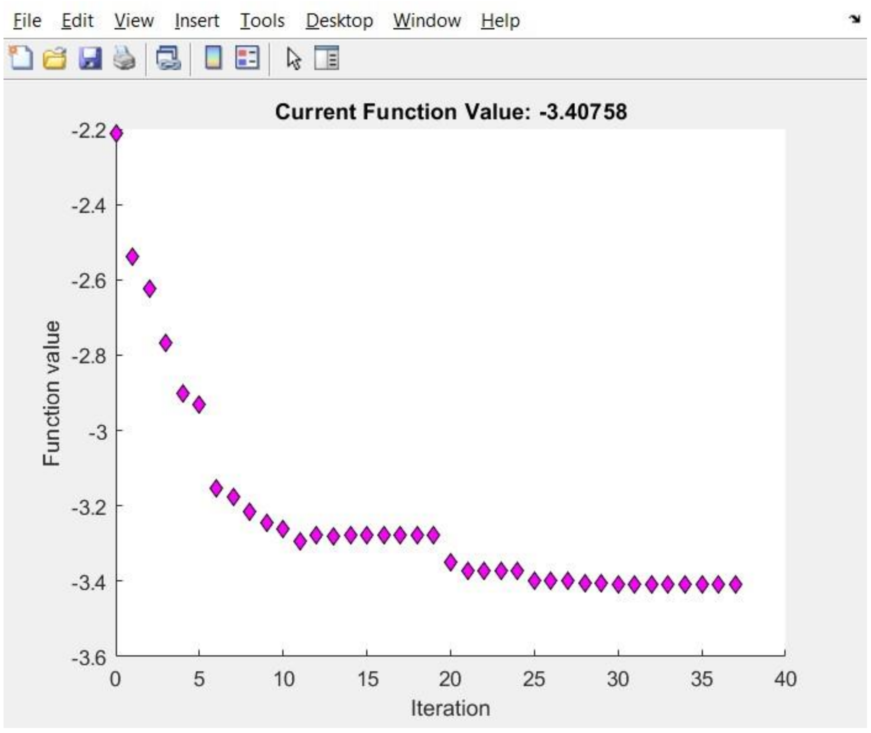
Results

```
w = 0.8378
r2 = 0.2748
alpha1 = 0.03000
Objective Value = 7.3059
Results difference: f/1.49 = 4.9033, 390.33% larger than nominal
```

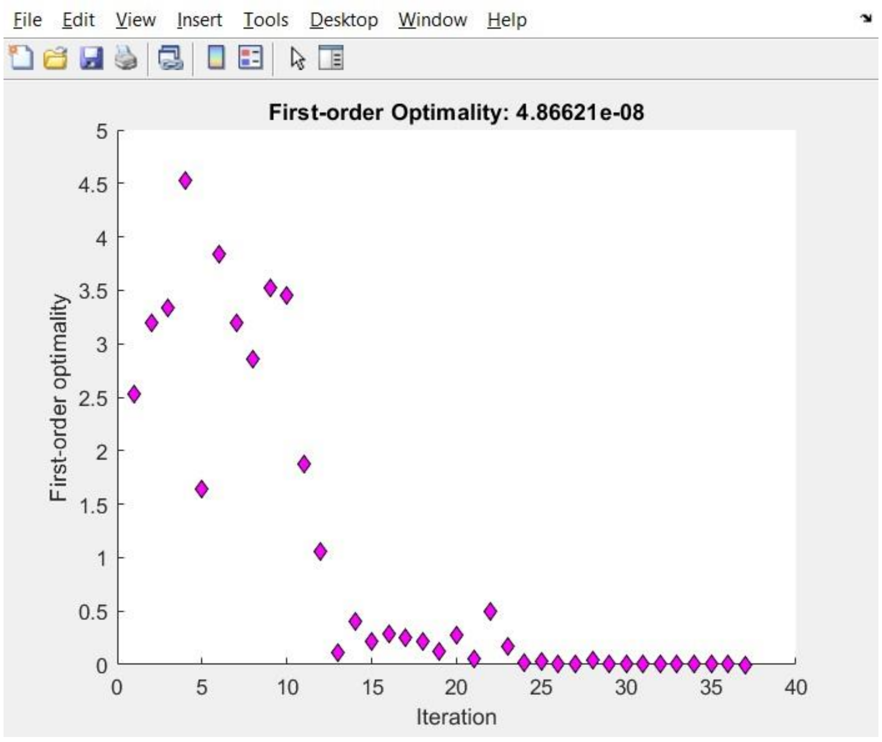
```
>> test_objective
At T = 5
objective function = 1.4964
Match with the results on Piazza
```

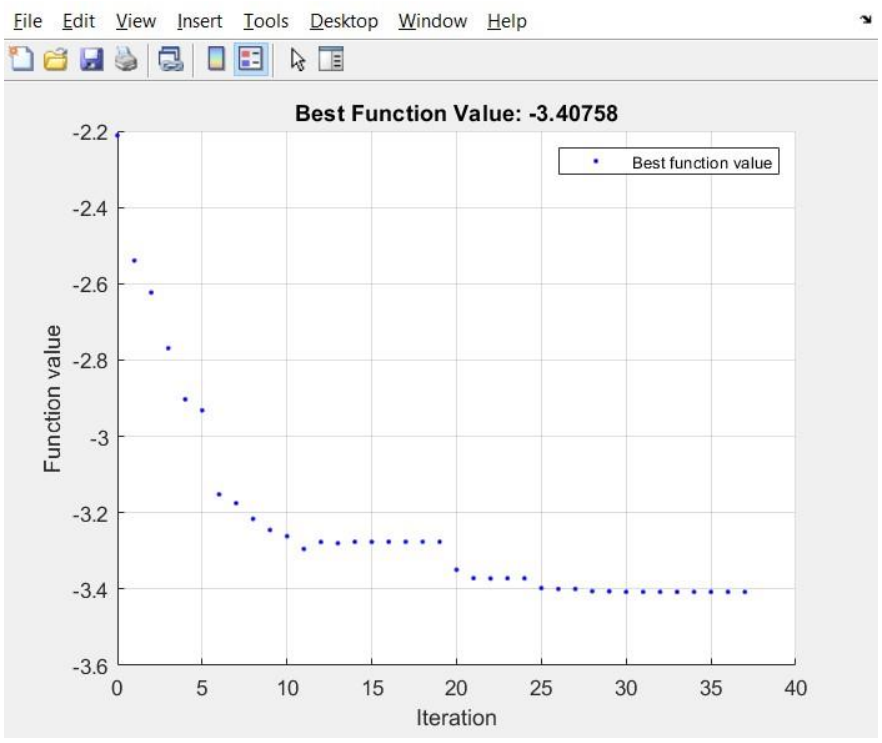
Integration period (T) is set to 5 and using nominal values ($\omega = 6.5 * \left(\frac{2\pi}{60}\right)\frac{rad}{s},\ r_2 = 0.8\ m, and\ \alpha_1 = 0.058\ rad$), we get an objective function of 1.4964. This result match with the value on Piazza.





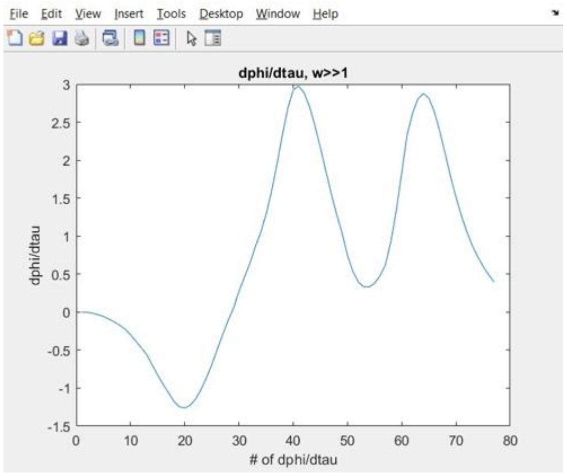
Convergence History

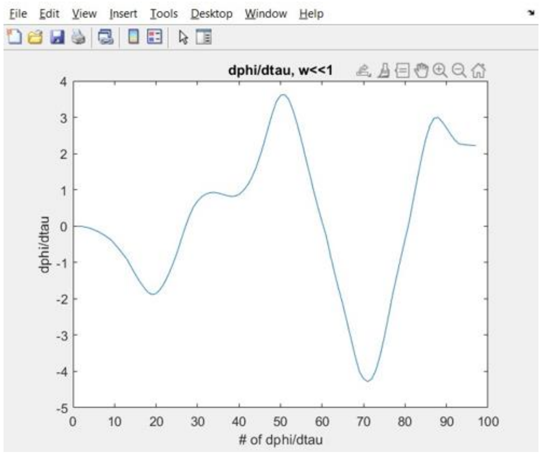




Conclusions

At both situations $\omega \ll 1$ or $\omega \gg 1$, ϕ and $\frac{d\phi}{dt}$ behave as expected. When ω is larger than 1 the angular velocity of the over all ride is fast, so the change in ϕ is slower comparing to situation when $\omega \gg 1$. The comparison between the plots are provided below.





Using 200 sample, T = 5, and three design variables(ω, r_2, α_1), we found a result of 7.3059, which is 390.33% larger than the nominal value(1.49). The final parameters are $\omega = 0.08378\, rpm, r_2 = 0.2748\, m, and\ \alpha_1 = 0.03000\, rad$. All variables are under constraints.

Appendix

A. Step_1_2

```
%% Step 1: Sample the 3D design space using
lhsdesign. This will produce
%            samples, but they will be between zero
and one.
% Design Variables:
%   w = theta_dot, 3 rpm <= w*(60/(2*pi)) <= 8 rpm
%   - angular velocity of the overall ride
%
%   r2, 0.1m <= r2 <= 1.5m
%   - the radial arm from the center of each car
platform to the center
%     of gravity of the car
%
%   alpha1, 0 rad <= alpha1 <= 0.3 rad
%   - defines the max/min incline angle of the beam
from the center
%     of the ride to the car platform

rng default % For reproducibility
% n: Number of samples
n = 200;
x = lhsdesign(n,3);

%% Step 2: Loop over the samples locations, and scale
and translate them so
%           that the design variables lie between the
correct lower and upper bounds (see note below about
alternative to scaling).

% Design variables
x1 = zeros(1,n);    % w = theta_dot, rpm, angular
velocity of overall ride.
x2 = zeros(1,n);    % r2, meter, the radial arm from
the center of each car platform to the center of
gravity of the car.
x3 = zeros(1,n);    % alpha1, rad, max/min incline
angle of the beam from the center of the ride to the
car platform.
for i = 1:n % w, rpm
    x1(i) = x(i) * (8*2*pi/60 - 3*2*pi/60) +
3*2*pi/60;
end
for i = 1:n % r2, m
```

```
    x2(i) = x(i) * (1.5-0.1) + 0.1;
end
for i = 1:n % alpha1, rad
    x3(i) = x(i) * (0.3-0) + 0;
end
```

B. GetMotion.m

```
function [tau,y1] = GetMotion(w, r2, alpha1)
% function [EoM] = GetMotion(w,r2,alpha1)
% Input:
%   w = theta_dot, 3 rpm <= w*(60/(2*pi)) <= 8 rpm
%   - angular velocity of the overall ride
%
%   r2, 0.1m <= r2 <= 1.5m
%   - the radial arm from the center of each car
platform to the center
%     of gravity of the car
%
%   alpha1, 0 rad <= alpha1 <= 0.3 rad
%   - defines the max/min incline angle of the beam
from the center
%     of the ride to the car platform
%
% Outputs:
%   tau = 3*w*t - nondimensional time
%   y(1),y(2) - solution to the ODE45
%-------------------------------------------------------
% Constants:
%   r1 = 4.3 m           - The beam length from the
center of the ride to
%                          the center of each platform
is fixed.
%   alpha0 = 0.036 rad   - incline offset; it controls
the incline at
%                          theta = 0
%   Q0 = 20              - Q0 = (m/p)sqrt(g(r2)^3),
%                            m - mass of the car
%                            rho - damping coefficient
to account for friction
%                            g - acceleration due to
gravity, m/s^2
r1 = 4.3;
alpha0 = 0.036;
Q0 = 20;
g = 9.81;

% dimensionless parameters
gama = 1/(3*w) * sqrt(g/r2);
```

```
epsilon = r1 / (9*r2);

%alpha = alpha0 - alpha1*cos(tau);
%beta = 3*alpha1*sin(tau);

tspan = [0 3*pi];
y0 = [0 0];
[tau, y] = ode45(@(tau,y) odefcn(tau,y,
gama,Q0,epsilon, alpha0, alpha1), tspan, y0);

function dydt = odefcn(tau, y,
gama,Q0,epsilon,alpha0,alpha1)
    dydt = zeros(2,1);
    dydt(1) = -(gama/Q0)*y(1)-(epsilon-
(gama^2)*(alpha0 - alpha1*cos(tau)))*sin(y(2))-
(gama^2)*(3*alpha1*sin(tau))*cos(y(2));
    dydt(2) = y(1);
end
y1 = y(:,1);
end
```

C. objective.m

```
%% Step3:
%           Loop over the scaled sample locations,
and evaluate the
%           objective function (this step could be
combined with step
%           2). At this point, you have the sample
locations and objective
%           values. These could be saved to file if
you want to save time
%           in the future
%-----------------------------------------------------
---------------------

function [f] = objective(w, r2, alpha1)
% Inputs:
%   Design Variables:
%   w = theta_dot, 3 rpm <= w*(60/(2*pi)) <= 8 rpm
%   - angular velocity of the overall ride
%
%   r2, 0.1m <= r2 <= 1.5m
%   - the radial arm from the center of each car
platform to the center
%     of gravity of the car
%
%   alpha1, 0 rad <= alpha1 <= 0.3 rad
%   - defines the max/min incline angle of the beam
from the center
```

```
%      of the ride to the car platform
% Outputs:
%    f - the standard deviation of dphi/dt over time
%
% Notes:

%Step_1_2
%w = x1;
%r2 = x2;
%alpha1 = x3;

% Iy1 - Integral of all y1
% T - the total(non-dimensional) period of simulation
n = size(w,1);
%disp(n)
Iy1 = zeros(1,n);
%T = zeros(1,n);
T = 5; %4; %5;
%tau = 0:.1:T;
for i = 1:(n)
    [tau, y1] = GetMotion(w(i), r2(i), alpha1(i));
    Iy1(i) = trapz(tau,y1);
    %for j = 1:size(tau,1)
    %     T(i) = T(i) + tau(j)/3/w(i);
    %end
end
% Calculte mean_y1 - mean angular velocity
%disp(T)
%disp(Iy1)
mean_y1 = Iy1./T;
%disp(mean_y1)

% objective values
f = zeros(1,n);
for i = 1:n
    [tau, y1] = GetMotion(w(i), r2(i), alpha1(i));
    f(i) = 3*w(i)* sqrt( trapz(tau,(y1-
mean_y1(i)).^2) / T );%T(i) );
end
%plot(y1);
%title("dphi/dtau, w>>1")
%xlabel("# of dphi/dtau")
%ylabel("dphi/dtau")
%f = -f;

plot(f)
```

D. surrogate.m

```
function [f] = surrogate(x)

% this is needed so Matlab knows where to find the
library if the gpml
% directory is not the working directory; set mydir
to the location of
% where you saved the gpml directory
%mydir = '~/Teaching/DesignOpt/gpml-matlab-v3.6/';
%addpath(mydir(1:end-1))
%addpath([mydir,'cov'])
%addpath([mydir,'doc'])
%addpath([mydir,'inf'])
%addpath([mydir,'lik'])
%addpath([mydir,'mean'])
%addpath([mydir,'prior'])
%addpath([mydir,'util'])

%Step_1_2
w = x(:,1);
r2 = x(:,2);
alpha1 = x(:,3);

%% define the function to be sampled
% fe = @(x) (x - 3).*(x.^3).*((x - 6).^4);
fe = objective(w, r2, alpha1);
%fe = @(x123) objective(x123(1,:), x123(2,:),
x123(3,:));

%% sample the function
% x = [1; 2; 3.5; 5.25; 7];
% y = fe(x);

%x123 = [sort(x1);sort(x2);sort(x3)]';
%x = [w(:) r2(:) alpha1(:)];
x = x;
y = fe(:);

%% set the squared exponential covariance function
%covfunc = @covSEiso; % {@covMaterniso, 1}; %
%hyp.cov = [log(1); log(10)]; % first component is
log(l) and second is log(sigma)
covfunc = {@covMaterniso, 1};
hyp.cov = log([1/4; 1.0]);


%% set the likelihood function to Gaussian
```

```
likfunc = @likGauss;
sn = 0.05; %1e-16; % this is the noise level
hyp.lik = log(sn);


%% maximize the likelihood function to find the
hyperparameters
hyp = minimize(hyp, @gp, -100, @infExact, [],
covfunc, likfunc, x, y);


%% now sample the GP for plotting purposes.  In
project 3, the variable z
% will be a single design provided to the objective
function by fmincon.
% Here, we want to plot the GP surrogate, so we need
to sample many z
% values.
%z = linspace(0, 7.5, 125)';
%[m s2] = gp(hyp, @infExact, [], covfunc, likfunc, x,
y, z);

[f s2] = gp(hyp, @infExact, [], covfunc, likfunc, x,
y);
f = -f;

%function [f] = surrogate(dv)
%     [f s2] = gp(hyp, @infExact, [], covfunc,
likfunc, dv, y); %stuff here needed by gp);
%     f = -f;
%end


%% ...and plot
%f = [m+1.96*sqrt(s2); flipdim(m-1.96*sqrt(s2),1)];
%fill([z; flipdim(z,1)], f, [7 7 7]/8)
%hold on;
%plot(z, m);
%plot(x, y, 'ks','MarkerFaceColor','k');
%plot(z, fe(z), 'k--')
%axis([0 7.5 -3000 3000]);

end
```

E. run_opt.m

```
%% run optimization using fmincon
```

6850
Project #3: tilt-a-whirl

```
% Displays:
%   Iter, F-count, f(x), Feasibility, First-order
optimality, Norm of step

% conducts the optimization
close all;
clear all;

% fmincon options
    %'optimplotfvalconstr' plots the best feasible
objective function value
    %found as a line plot. The plot shows infeasible
points as red and feasible
    %points as blue, using a feasibility tolerance of
1e-6.
options =
optimoptions('fmincon','Display','iter','PlotFcns',@o
ptimplotfvalconstr);%,@optimplotfval
@optimplotfirstorderopt); %,);
%loglog('optimplotfirstorderopt')

%define initial design, w, r2, alpha1
%Step_1_2
%x0 = [x1;x2;x3]';
x0 = [0.5792; 0.9689; 0.6405]';
%x0 = [0.5792; 0.9689; 0.6405]';

% define obj function
%fun = @(a)objective(a, L, Nx, Ny, kappa, Ttop,
Tbot);
fun = @(x)surrogate(x);

% define Aineq, bineq
lb = [3*2*pi/60, 0.1, 0];
ub = [8*2*pi/60, 1.5, 0.3];
A = [];
b = [];
Aeq = [];
beq = [];

% call fmincon
[a, f] =
fmincon(fun,x0,A,b,Aeq,beq,lb,ub,[],options);

% plot/display results
%plot(Iter, 1st_Order_Optimality)
%plot(Iter, Feasibility)
```

F. Tests

F.1. test_GetMotion.m

```
% test GetMotion
clear all;

%% one value
% Design variables
w = 6.5*(2*pi/60);
r2 = 0.8;
alpha1 = 0.058;
[tau, y1] = GetMotion(w, r2, alpha1);

%% Use Step_1_2 TrapzValues
% Design variables
%Step_1_2
%w = x1;
%r2 = x2;
%alpha1 = x3;
% Nx - Number of samples
%TrapzValues = zeros(1,n);
%for i = 2:(n)
%     [tau, y1] = GetMotion(w(i), r2(i), alpha1(i));
%     TrapzValues(i) = trapz(tau,y1);
%end
```

F.2. test_gp.m

```
%% Test 1
%%%clear all;

%%%Step_1_2
%%%x123 = [x1;x2;x3]';
%y = fe(x123);

%%%f = objective(x123(:,1), x123(:,2), x123(:,3));
%%%disp(f);

%% Test 2
% startup script to make Octave/Matlab aware of the
GPML package
%
% Copyright (c) by Carl Edward Rasmussen and Hannes
Nickisch 2014-11-30.
```

```
disp(['executing gpml startup script...']);

OCT = exist('OCTAVE_VERSION') ~= 0;           % check
if we run Matlab or Octave

me = 'Code gpml-matlab-v3.6-2015-07-
07';                                                    %
what is my filename
mydir = which(me); mydir = mydir(1:end-2-
numel(me));        % where am I located
if OCT && numel(mydir)==2
  if strcmp(mydir,'./'), mydir = [pwd,mydir(2:end)];
end
end                   % OCTAVE 3.0.x relative, MATLAB
and newer have absolute path

addpath(mydir(1:end-1))
addpath([mydir,'cov'])
addpath([mydir,'doc'])
addpath([mydir,'inf'])
addpath([mydir,'lik'])
addpath([mydir,'mean'])
addpath([mydir,'prior'])
addpath([mydir,'util'])

clear me mydir


clear all;

Step_1_2
x123 = [x1;x2;x3]';
%y = fe(x123);

f = objective(x123(:,1), x123(:,2), x123(:,3));
disp(f);

covfunc = {@covMaterniso, 1};
hyp.cov = log([1/4; 1.0]);
```

F.3. test_objective.m

```
clear all;

%% Test 1
%Step_1_2
%%Step_1_2
%%w = x1;
```

```
%%r2 = x2;
%%alpha1 = x3;
%[f] = objective(sort(x1),sort(x2),sort(x3));
%plot(sort(x1),f)


%% Test 2
%%%Step_1_2

% define the function to be sampled
% fe = @(x) (x - 3).*(x.^3).*((x - 6).^4);

% sample the function
% x = [1; 2; 3.5; 5.25; 7];
% y = fe(x);
%w = x1;
%r2 = x2;
%alpha1 = x3;
%%%x123 = [sort(x1);sort(x2);sort(x3)];

%%%f = objective(x123(1,:), x123(2,:), x123(3,:));
%% Test 3 - Nominal Value
% Select a suitable value for T
% objective value in range[1.49,1.61]. - Nominal
Value
%w = 6.5*(2*pi/60);
%r2 = 0.8;
%alpha1 = 0.058;
%f = objective(w,r2,alpha1);
%disp("At T = 5")
%disp("objective function = " + f)
%disp("Match with the results on Piazza")

w = 0.8378;
r2 = 0.2748;
alpha1 = 0.03;
f = objective(w,r2,alpha1);
```

F.4. test_objective.m

```
%% Test surrogate()

clear all;
close all;

Step_1_2
x = [x1' x2' x3'];
f = surrogate(x);
```