Project 3 – MANE 6710 Numerical Design Optimization

Unique ID: 5284

Table of Contents

INTRODUCTION **2**

DETERMINISTIC CHAOS **2**

Simulating Deterministic Chaos 4

OBJECTIVE FUNCTION **5**

Choice of Design Variables 5

Formulation of Objective function 5

SURROGATE MODEL **6**

Accuracy of Surrogate Model 7

OPTIMIZATION SETUP **9**

RESULTS **9**

Convergence Analysis 10

WORKS CITED **11**

List of Figures

Figure 1. Top View of Tilt-A-Whirl platform 2

Figure 2. Various Orientations of Tilt-A-Whirl design 3

Figure 3. a) Starting point $[\pi 3,0]$, b) Starting point $[\pi 5]$ 4

Figure 4. 1-D slice of objective function along omega 6

Figure 5. Surrogate model vs True Objective comparison 7

Figure 6. Mean Square Error for 1-D slice of objective v Surrogate 8

Figure 7. tau = 500, bins = 1000 surrogate performance 8

Figure 8. First Order Optimality plot 10

Figure 9. Convergence Analysis plot 10

Figure 10. Angular velocity comparison between starting and optimal point 10

List of Tables

Table 1. Variable Description 3

Table 2. a) Constant parameters. b) Design Variables and their range 5

Table 3. Trained Hyperparameters 6

Table 4. Physical Intuition of results 9

Introduction

The main objective of this project is to achieve optimal parameters to design Tilt-A-Whirl. Tilt-A-Whirl is a theme park joy ride which has its physics based on deterministic chaos. People going on this ride have a lot of fun because they cannot anticipate their motion throughout the ride. This project focusses on achieving design parameters of Tilt-A-Whirl mechanism that can maximize this chaotic behavior, enabling people taking the ride to have utmost fun.

Deterministic Chaos

It is important to provide an explanation about deterministic chaos before further work is discussed in detail. A dynamical system expressed as Ordinary Differential Equations (ODEs), is the rate of change of certain "state" variables with respect to time governed by underlying physics and deterministic laws. This dynamical system becomes chaotic if there is no way to predict what or how the state variables will behave over a long period of time. Tilt-A-Whirl extracts all its fun elements from this very concept of unpredictability.

Tilt-A-Whirl design [1] is parametrized by several variables, and it is designed as shown in Figure 1.

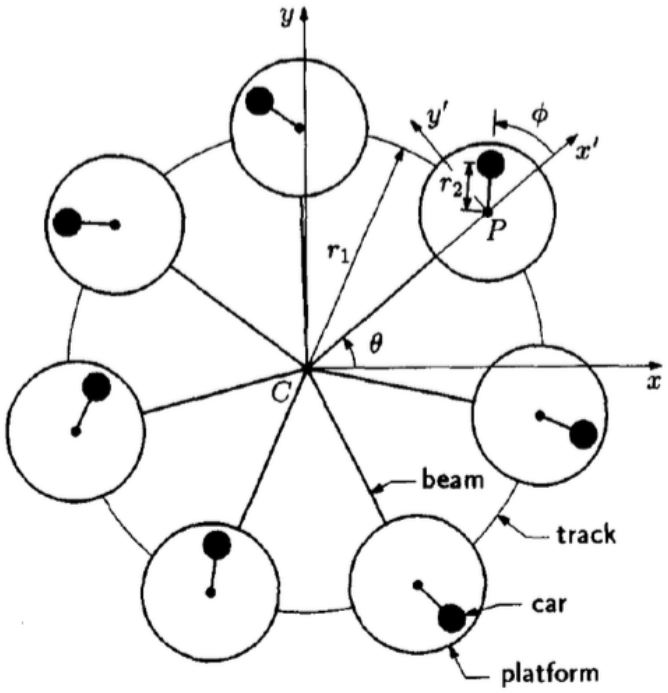


Figure 1. Top View of Tilt-A-Whirl platform

Work done in [1], derives the equations of motion for the system described in Figure 2. Various Orientations of Tilt-A-Whirl design and the variables used are described in Table 1.

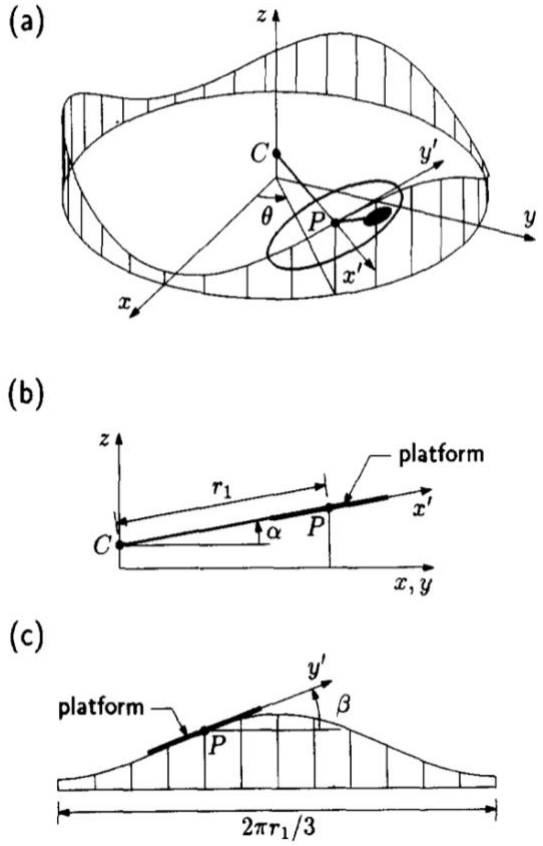


Figure 2. Various Orientations of Tilt-A-Whirl design

Variable	Description	Units
r_1	Distance from axis to center of platform	m
r_2	Distance from center of platform to car	m
θ, ω	Location of platform and angular velocity	rad, 1/s
$\phi, \frac{d\phi}{dt}$	Position of car and its angular velocity	rad, 1/s
ϵ	$r_1/(9r_2)$	No dim
γ	Non dimensional term	No dim
$\{x, y, z\}$	Global coordinate system	NA
$\{x', y', z'\}$	Local coordinate system of each car	NA
α, β	Angles of (x', y') w.r.t (x, y)	rad
τ	$3\omega t$, t - seconds	No dim

Table 1. Variable Description

The equation of motion is:

$$\frac{d^2\phi}{d\tau^2} + (\gamma/Q_0)\frac{d\phi}{d\tau} + (\epsilon - \gamma^2\alpha)\sin\phi + \gamma^2\beta\cos\phi = 0 \qquad \textit{Eqn 1}$$

$$\epsilon = r_1/(9r_2) \qquad \textit{Eqn 2}$$

$$\alpha = \alpha_0 - \alpha_1 \cos\tau \qquad \textit{Eqn 3}$$

$$\beta = 3\alpha_1 \sin\tau \qquad \textit{Eqn 4}$$

$$\gamma = (1/3\omega)\sqrt{g/r_2} \qquad \textit{Eqn 5}$$

$$Q_0 = (m/\rho)\sqrt{gr_2^3} \approx 20 \qquad \textit{Eqn 6}$$

Here, m – mass and ρ is the damping constant. $Q_0 \approx 20$ ensures the system is over-damped making the user experience fun as well as safe.

Simulating Deterministic Chaos

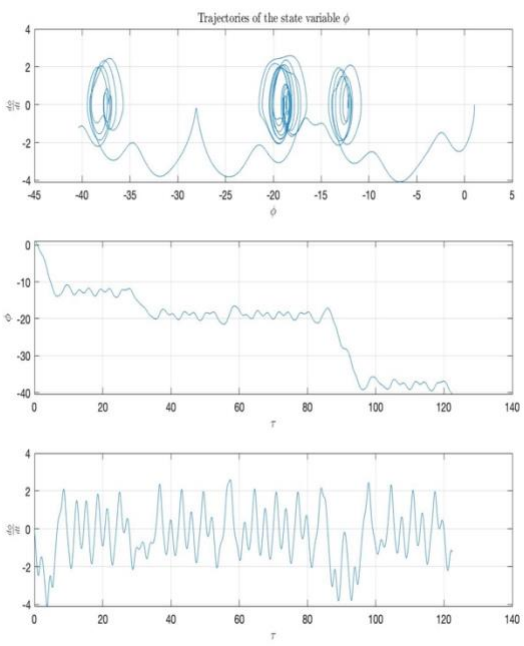
The next step is to simulate Eqn 1, using the state-space method of re-writing this equation.

$$\begin{gathered} y_1 = \phi \\ y_2 = \dot{y_1} = \dot{\phi} \\ \dot{y_2} = \ddot{y_1} = \ddot{\phi} \end{gathered}$$

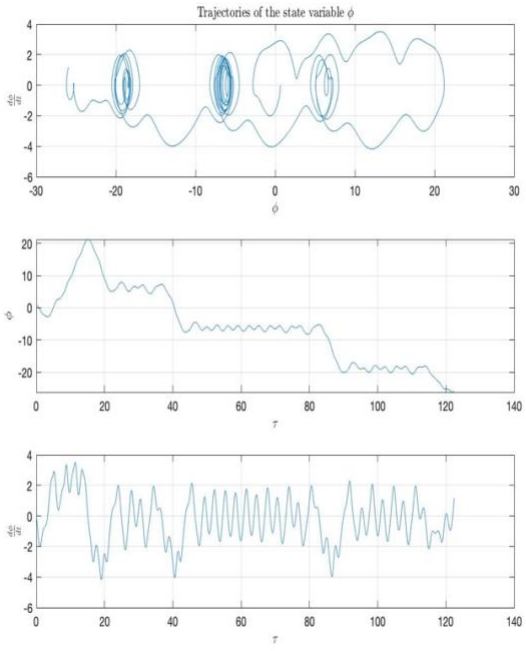
$$\begin{bmatrix} \dot{y_1} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} y_2 \\ -((\gamma/Q_0)y_2 + (\epsilon - \gamma^2\alpha)\sin y_1 + \gamma^2\beta\cos y_1) \end{bmatrix} \qquad \textit{Eqn 7}$$

This form of the equation as derived in Eqn 7 can be simulated using MATLAB's ODE simulation function *ODE45(@chaosDynamics, Time Range, Starting point).* This command takes function handle @chaosDynamics (Eqn 7), the time range τ (non-dimensional time) and the starting point of the states $\left[\phi, \frac{d\phi}{d\tau}\right]$. For starting points $\left[\frac{\pi}{3}, 0\right]$ and $\left[\frac{\pi}{5}, 0\right]$ and a non-dimensional time $\tau \in [0,125]$ the states vary as shown in Figure 3. ($r_2 = 0.8m, \alpha_1 = 0.058\, rad, \omega = 6.5\, rpm$)

Figure 3. a) Starting point $[\pi/3\,, 0]$*, b) Starting point* $[\pi/5]$



a)



b)

Table 2. shows the list of parameters which are fixed as design constraints and those that cannot be varied. From Figure 3., it can be clearly seen that for different starting points and these set of parameters, the trajectories of $\{\phi, \frac{d\phi}{dt}\}$, vary significantly over time. This is because of the chaotic nature of the dynamical system. The most important point to be noted is the noisy nature of how $\frac{d\phi}{dt}$ varies with respect to time.

Table 2. a) Constant parameters.

Parameters	Values	Units
r_1	4.3	m
α_0	0.036	rad
g	9.81	m/s^2

b) Design Variables and their range

Design Vars	low	high
r_2	0.1 m	1.5 m
α_1	0 rad	0.3 rad
ω	3 rpm	8 rpm

$\frac{d\phi}{dt}$ is the angular velocity of the car and passengers will have fun if it gets noisier.

Objective function

Choice of Design Variables

Table 2.a) lists all the parameters which are constant from a design perspective. So, parameters r_2, α_1, ω are the remaining variables which can be used to modify the behavior of $d\phi/dt$ with respect to t. These are the choice of design variables in this project. Table 2.b) lists the design variables and their permissible range of values.

Formulation of Objective function

The objective is to have maximum possible chaotic behavior in $d\phi/dt$ which means, it is important to achieve high variance or standard deviation of the trajectory of $d\phi/dt$ over large intervals of time.

$$f(r_2, \alpha_1, \omega) = \sigma(d\phi/dt\ )$$

$$\sigma(y_2) = \sqrt{\frac{1}{T}\int_0^T (y_2 - \overline{y_2})^2\ dt} \qquad \text{Eqn 8}$$

$$\overline{y_2} = \frac{1}{T}\int_0^T y_2\ dt$$

Where, T is the total simulation time and $y_2 = d\phi/dt$. A one-dimensional slice of this objective function with fixed $r_2 = 0.8\,m, \alpha_1 = 0.058\,rad$ and varying ω over its permissible range of values is shown in Figure 4. This is proof of how noisy the actual objective function is. A gradient based optimizer will find it difficult to get to an optimal point due to the extremely spiky objective function and it will get lost in one of its multiple local minimizers. Hence, it is important to get a smooth representation of the objective function in order to find an optimal point.

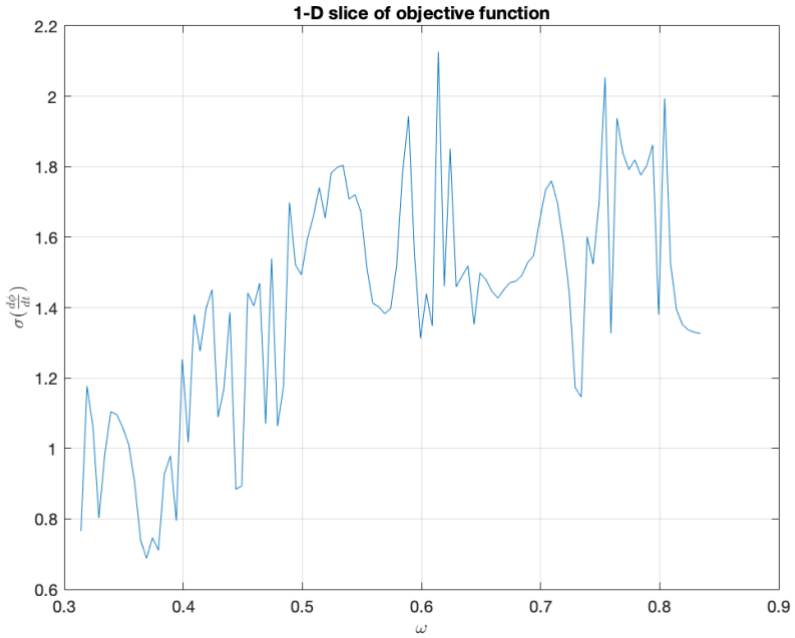


Figure 4. 1-D slice of objective function along omega

Surrogate Model

The next step in this project is to generate a surrogate model that presents a smooth representation of this noisy objective function behavior. Gaussian Process Library (GPML) is used to generate the surrogate model. Sample points are generated by the Lattice Hypercube Sampling technique and the true objective function values are computed for each of those sample points. The true objective values at these sample locations are labelled as the training data and it drives the process of generating a Gaussian process surrogate. 1000 points have been sampled within the range of permissible values of the design variables. $gp(hpy, @infExact, [\ ], covfunc, likfunc, x, y, z)$ is the function from GPML library which can be used to perform prediction using the trained parameters. The trained hyperparameters ($\tau = 50, bins = 1000$) are shown in Table 3.

Table 3. Trained Hyperparameters

Covariance	$[1.8094 \quad 1.3928]$
Likelihood	-2.1935

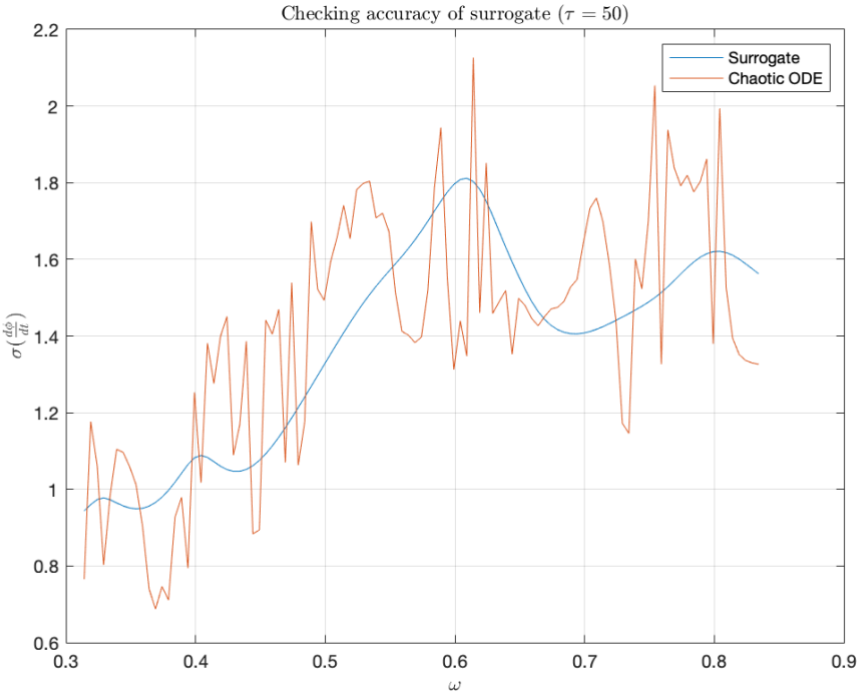


Figure 5. Surrogate model vs True Objective comparison

Figure 5. shows how the surrogate prediction of the 1-D slice of the objective function compared to the true objective function. It can be seen that the prediction is a smooth representation of the chaotic, noisy true objective function. The surrogate can be used as an alternative to the true objective function in the optimization process if it is an accurate enough approximation.

Accuracy of Surrogate Model

It is important to have an accurate surrogate that represents the true objective function or else the gradient based optimizer will lead to an incorrect optimal point. The accuracy is checked by computing the mean squared error between the prediction and the true objective of the 1-D slice in consideration.

$$\frac{1}{\omega_2 - \omega_1}\int_{\omega_1}^{\omega_2}\left(\hat{\dot{\phi}} - \dot{\phi}\right)^2 d\omega \approx \frac{1}{N}\sum_{i=1}^{i=N}\left(\hat{\dot{\phi}} - \dot{\phi}\right)^2 \qquad \textit{Eqn 9}$$

To do this analysis, a lot of variables can be used to compute the accuracy of the surrogate, but the choice of variables is number of sample data and the integration time period of the differential equation.

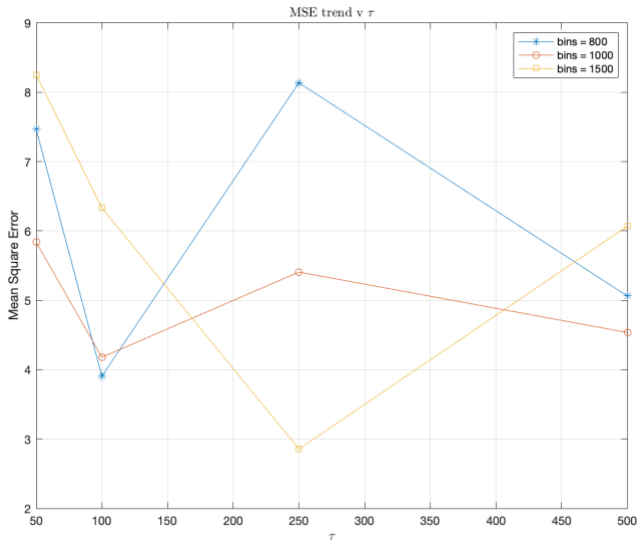


Figure 6. Mean Square Error for 1-D slice of objective v Surrogate

The least error between the surrogate and the true objective function was observed when the number of bins used were 1500 and a total non-dimensional integration time of the chaotic differential equation is $\tau = 250$. But this is a very small integration period to truly enjoy the chaotic nature of the ride. For a maximum non-dimensional integration time of $\tau = 500$, the least MSE was observed when the number of bins used in LHS was 1000. The fit of surrogate model with the true objective for, $\tau = 500, lhs\ bins = 1000$, is shown in Figure 7.

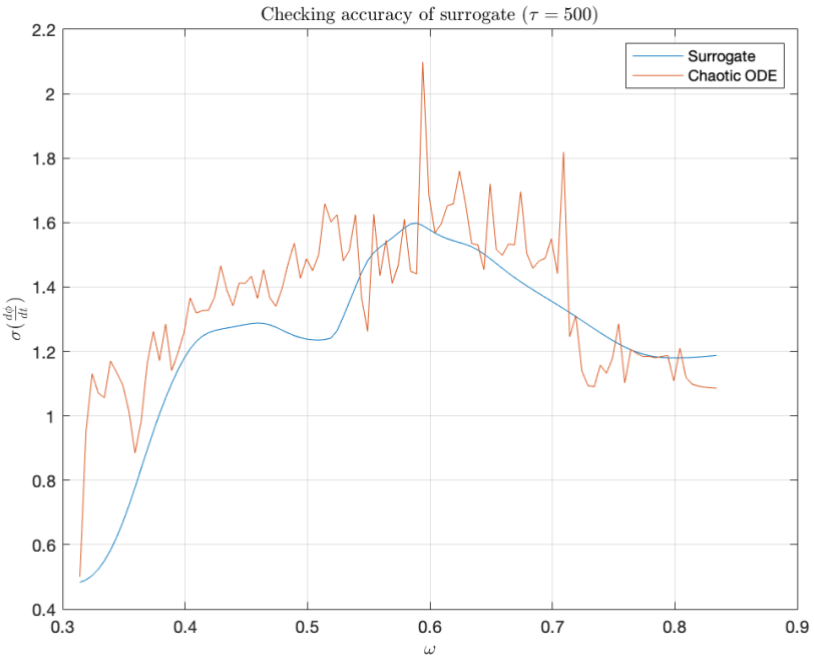


Figure 7. tau = 500, bins = 1000 surrogate performance

Optimization setup

With the surrogate model up and setup, the optimization statement can be written as:

$$\max_x \sigma\left(\frac{d\phi}{dt}\right) \approx \max_x \hat{\sigma}\left(\frac{d\phi}{dt}\right) = \min_x -\hat{\sigma}\left(\frac{d\phi}{dt}\right) \qquad \textit{Eqn 10}$$

where, $x - \{r_2, \alpha_1, \omega\}$ and the smooth surrogate approximation is used as the objective function. The range of the design variables are the lower and upper bound constraints. MATLAB's inbuilt ***fmincon(obj,x0,A,b,Aeq,beq,lb,ub,nonlincon,opt)*** is used for performing the optimization and find the optimal maximizer.

Results

For a starting point of $\{r_2, \alpha_1, \omega\}_0 = \{0.8, 0.058, 6.5\}$ and a non-dimensional integration period of $\tau = 500$, the objective at the starting point is 1.399723256398801. A surrogate trained with $lhs\ bins = 1000$ for the same τ, the optimal point achieved is
$\begin{bmatrix} 0.103345311283907 \\ 0.295620715207665 \\ 0.792497899077308 \end{bmatrix}$.
The **optimal standard deviation achieved using the surrogate is 6.533553656835124**.
This result translates into this following physical intuition of the problem statement described in Table 4

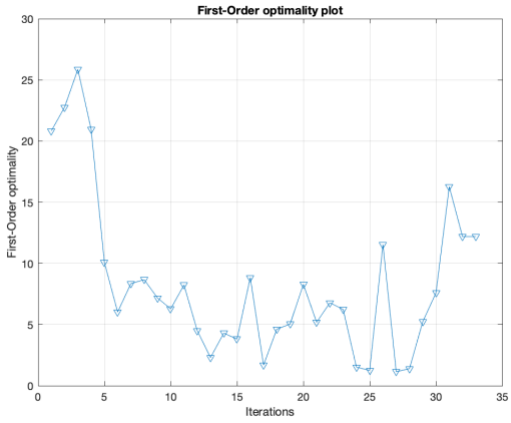
Table 4. Physical Intuition of results

Variable	Initial point	Optimal point	Physical Intuition
r_2, m	0.8	0.103345311283907	Decrease in r_2, is a rise in ϵ, which implies a reduction of torque required to spin the car around on the platform
α_1, rad	0.058	0.295620715207665	Increase in α_1 directly affects the angles α, β (angle of $\{x', y'\}$). It makes the ride a lot bumpier compared to the flatter initial design.
ω, rpm	6.5	7.567797481685735	A rise in ω is a clear indication of the fact that, a faster ride has direct correlation to increased fun.

Optimal standard deviation achieved using ODE45 is 6.616275299155537.This is a result which aligns with the physical intuition of the system dynamics and shows an improvement of approximately 366 % from the original objective. This also shows that the surrogate is a good approximation for the real deterministic chaotic simulation.

The First-Order optimality plot shown in Figure 8. is not a traditional looking plot because, the surrogate objective is trying fit the true objective which is extremely noisy. This makes the surrogate also a little noisy and that is reflected in Figure 8.

Figure 8. First Order Optimality plot



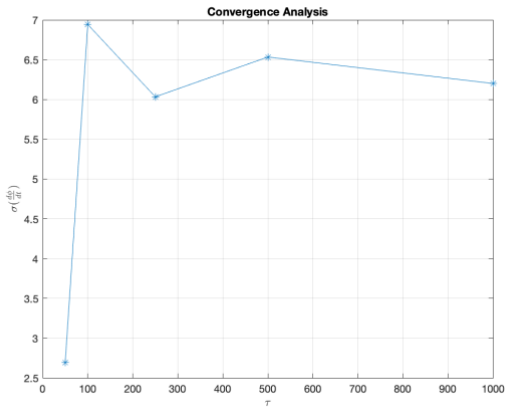


Figure 9. Convergence Analysis plot

Convergence Analysis

Increasing the non-dimensional integration time is equivalent to increasing the integration time period of the ODE as they are directly proportional to each other. With an increase in the integration period, a rise in the objective value is noticed and it converges around the band of $(6,7)$ values of standard deviation. Since the true objective is a noisy function, the optimal function values converging to a band will not be a surprise given its chaotic nature. This can be clearly seen in Figure 9.

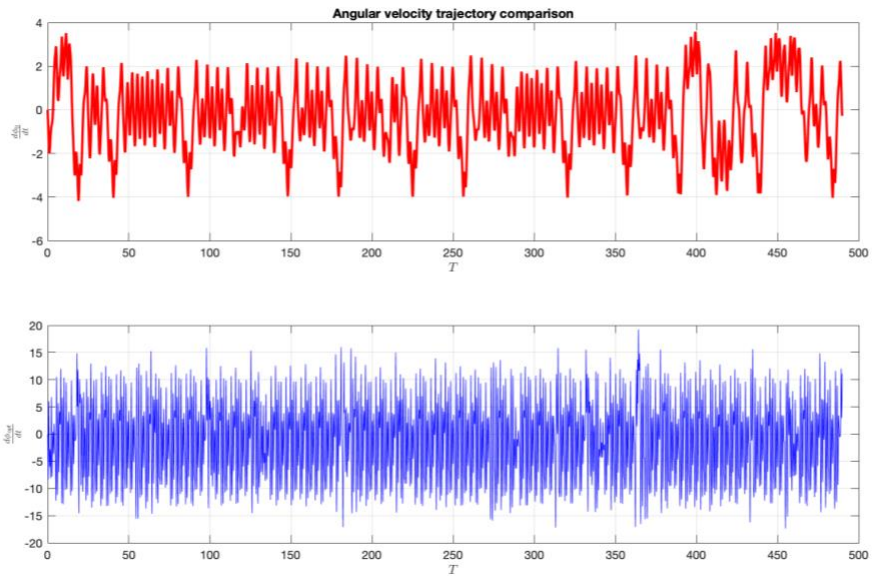


Figure 10. Angular velocity comparison between starting and optimal point

The red curve is the trajectory of $\dot{\phi}$ with initial parameters. The blue curve is the trajectory of $\dot{\phi}$ with optimal design parameters.

Works Cited

[1] B. M. R.L.Kautz, "Chaos at the amusement park: Dynamics of the Tilt-A-Whirl," *American Association of Physics Teachers*, vol. 62, no. 1, pp. 59-65, 29 June 1993.

Contents

- sample the function
- train gaussian process surrogate
- Comparison of surrogate model with Chaotic ODE
- plot
- save variables


```
clc
clear all;
```

```
mydir = '/Users/vignesh/Desktop/RPI/Fall 2021/MDO/Project_3/GPML_OPT_code';
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'])
```

sample the function

```
% r_2
r_2_in = 0.1;
r_2_out = 1.5;

% \alpha_1
alpha1_in = 0;
alpha1_out= 0.3;

% \omega
omega_in = 2*pi*3/60;
omega_out = 2*pi*8/60;

% Range vector
% [ r_2(in) r_2(out)]
% [ alp(in) alp(out)]
% [ omg(in) omg(out)]
Range = [r_2_in r_2_out;alpha1_in alpha1_out;omega_in omega_out];


% sampling process
bins = 1000;
dim = length(Range);
x = samplingProject3(Range,bins,dim);

% true objective generation from sampled data points
tau_int = 1000;
tau = 0:0.01:tau_int;
y = zeros(length(x),1);
Y0 = [pi/3;0];

for i=1:length(x)
```

```
    y(i,1) = ODEsim(Y0,tau,x(i,:));
end
```

train gaussian process surrogate

```
% set the squared exponential covariance function
covfunc = {@covMaterniso,1};  % @covSEiso
hyp.cov = [log(0.3); log(1)]; % first component is log(l) and second is log(sigma)

% set the likelihood function to Gaussian
likfunc = @likGauss;
sn = 0.25; %1e-16; % this is the noise level
hyp.lik = log(sn);

% maximize the likelihood function to find the hyperparameters
hyp = minimize(hyp, @gp, -300, @infExact, [], covfunc,...
    likfunc, x, y);
```

Comparison of surrogate model with Chaotic ODE

```
% Generate a 1-D slice of the function along \omega
z1 = 0.8;  % r_2 - constant
z2 = 0.058; % \alpha_1 - constant
Om = omega_in:0.005:omega_out; % range of omega

z = [ones(length(Om),1)*z1 ones(length(Om),1)*z2 Om'];
% surrogate approximates of the true objective (1-D slice)
[m s2] = gp(hyp, @infExact, [], covfunc, likfunc, x, y, z);

m2 = zeros(length(Om),1);
for i=1:length(Om)
    % True objective values along the 1-D slice
    m2(i) = ODEsim(Y0,tau,z(i,:));
end

% computing surrogate error
surrErr = compareSurrogate(m,m2,Om);
```

plot

```
figure
plot(Om,m); % surrogate model
hold on;
grid on;
plot(Om,m2); % Chaotic ODE
xlabel('$\omega$','Interpreter','latex');
ylabel('$\sigma(\frac{d\phi}{dt})$','Interpreter','latex');
legend('Surrogate','Chaotic ODE');
title('Checking accuracy of surrogate'...
    ,'Interpreter','latex');

figure
plot(Om,m2);
grid on
xlabel('$\omega$','Interpreter','latex');
ylabel('$\sigma(\frac{d\phi}{dt})$','Interpreter','latex');
title('1-D slice of objective function');
```

save variables

```
fname = input('Enter file name:','s');
save(fname,'x','y','Range','tau','hyp');
```

Published with MATLAB® R2021a

```
function samples = samplingProject3(Range,bins,dim)

    % use Lattice Hypercube Sampling to generate samples
    t = lhsdesign(bins,dim);

    % Scale design variables to their specified range
    for i=1:dim
        t(:,i) = Range(i,1) + (Range(i,2)-Range(i,1)).*t(:,i);
    end

    samples = t;

end
```

Published with MATLAB® R2021a

```
function I = ODEsim(Y0,tau,X)
% X(1) - r_2 value - m
% X(2) - \alpha_1 value - rad
% X(3) - \omega value - rad/s

[~,YP] = ode45(@(t,Y)chaos_subfn(t,Y,X),tau,Y0);

% YP(:,1) - holds \phi values
% YP(:,2) - holds d\phi/d\tau values

% d\phi/dt = ( d\phi/d\tau ) * ( d\tau/dt)
% \tau = 3 * X(3) * t
% d\tau/dt = 3* X(3)

dphi_dt = YP(:,2) * 3 * X(3);

% T - range of time (T = tau / ( 3 * X(3) ))
T = tau./ (3 * X(3));

% compute mean of dphi_dt - (1/T_end) * integral ( dphi_dt dt)
mean_dpdt = (1/T(end))*trapz(T,dphi_dt);

% squared difference calculation
sq_diff = (dphi_dt-mean_dpdt).^2;

% Final Integral calculation
I = sqrt((1/T(end))*trapz(T,sq_diff));


    function Ydot = chaos_subfn(t,Y,X)
        Q0 = 20;
        r1 = 4.3;
        g = 9.81;
        alpha0 = 0.036;

        omega = X(3);
        alpha1 = X(2);
        r2 = X(1);

        e = r1/(9*r2);
        Gamma = (1/(3*omega))*sqrt(g/r2);
        alpha = alpha0-alpha1*cos(t);
        beta = 3*alpha1*sin(t);

        Ydot(1) = Y(2);
        Ydot(2) = -(e-Gamma^2*alpha)*sin(Y(1)) - ...
            Gamma^2*beta*cos(Y(1))-(Gamma/Q0)*Y(2);
        Ydot = Ydot';
    end
end
```

Published with MATLAB® R2021a

Contents

- Load Surrogate parameters
- Set Initial point and run optimizer


```
clc
clear all;
```

Load Surrogate parameters

```
file = uigetfile('.mat');
load(file);

Range = evalin('base','Range');
lb = Range(:,1);
ub = Range(:,2);
```

Set Initial point and run optimizer

```
x0 = [0.8;0.058;2*pi*6.5/60];

options = optimoptions('fmincon','Display','iter-detailed','Algorithm','sqp',...
    'MaxIterations',1500);
[x_opt,fval] = fmincon(@obj,x0,[],[],[],[],lb,ub,[],options);
```

Published with MATLAB® R2021a

Objective function calculation

```
function f = obj(x)
    % x - design variable
    f = subObj(x);

    function fun = subObj(z)

        % set the squared exponential covariance function
        covfunc = {@covMaterniso,1};
        % trained hyper parameters - read from Workspace
        hyp_t = evalin('base','hyp');

        % set the likelihood function to Gaussian
        likfunc = @likGauss;

        % read training data
        % x - training points; y - training obj function
        x_t = evalin('base','x');
        y_t = evalin('base','y');

        fun = gp(hyp_t, @infExact, [], covfunc, likfunc, x_t, y_t, z');

        % negate fun value to find the optimal point that generates
        % maximizer
        fun = -1 * fun;
    end

end
```

Published with MATLAB® R2021a