MANE 6710 - Numerical Design Optimization Lab 3

Human 6966

November 15 2024

Table of Contents:

Executive Summery

Tilt-A-Whirls are a common and beloved small-town fair ride suitable for children and adults alike. Their fun comes from the random changes in the spin direction of the cart the rider is in as the ride goes over a hilly track. This behavior is due to the chaotic dynamics of the underdamped double pendulum that governs the ride's dynamics. The task of this lab is to maximize the chaotic nature of the ride by setting the second arm length, speed, and hill size using a surrogate model based optimization scheme in Matlab. The dynamics of the ride are analyzed through a non-dimentionalized equation from a paper by R. L. Kautz and B. M. Huggard. The GPLM surrogate model optimized by fmincon() was successfully able to maximize the standard deviation of the cart's angular velocity at state conditions.

1 Analysis Introduction

1.1 Methodology

The dynamics model used for the behavior of the Tilt-A-Whirl is given on the 63rd page of "Chaos at the amusement park: Dynamics of the Tilt-A-Whirl" by R. L. Kautz and B. M. Huggard and is shown in equation 1 [1].

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

With:

1. ϕ being the angular position of the center of mass about the axis of rotation of the cart
2. τ being nondimensional time
3. Q_0 being the energy dissipation ratio of the cart per rotation
4. $\epsilon = \frac{r_1}{9r_2}$
5. $\alpha = \alpha_0 - \alpha_1 * \cos(\tau)$
6. $\beta = 3\alpha_1 \sin(\tau)$
7. $\gamma = \frac{\sqrt{g}}{3\omega\sqrt{r_2}}$
8. r_1 being the radius between the ride's and cart's axies of rotation
9. r_2 being the radius between the center of mass and the axis of rotation of the cart
10. α_0 being the initial slope of the hills
11. α_1 being the incline angle of the hills
12. g being gravitational acceleration constant (9.81 m/s/s)
13. ω being the angular velocity of the ride about its axis of rotation

Equation 1 is converted into the state space representation given in equation 2 so the equation could be solved using Matlab's ode45() function.

$$\dot{\vec{\phi}} = \begin{bmatrix} \dot{\phi_1} \\ \dot{\phi_2} \end{bmatrix} = \begin{bmatrix} \phi_2 \\ -1(\frac{\gamma}{Q_0}\phi_2 + (\epsilon - \gamma^2\alpha)\sin(\phi_1) + \gamma^2\beta\cos(\phi_1)) \end{bmatrix} \tag{2}$$

With states $\phi_1 = \phi$ and $\phi_2 = \frac{d\phi}{d\tau}$ (Note the .-1 was factored out of the second state equation to reduce the error of the floating point math as it would accumulate and affect results). This state space system can be solved using a nonlinear differential equation solver like ode45(), the results of which can be used in the objective function using equation 3.

$$\sigma_{\dot{\phi}} = -3\omega\sqrt{\frac{\int_0^T (\phi_2 - \bar{\phi_2})^2 d\tau}{T}} \tag{3}$$

Where:

1. T is the nondimensional period of the ride
2. σ_{dotphi} is the standard deviation of the cart's angular velocity
3. $\bar{\phi_2} = \frac{\int_0^T \phi_2 d\tau}{T}$

Note that since Matlab's fmincon() is a minimization function and the task is to maximize the standard deviation of the cart's angular velocity, the objective equation (equation 3) results in -1* the standard deviation of the cart's angular velocity. Additionally Matlab's trapz() was used to compute the numerical integration.

1.2 Assumptions

To simplify the physical model and fundamental equations, the following assumptions were made:

1. Q_0, the term that accounts for mass and friction, has a constant value of 20
2. Negligible air resistance
3. The ride's structure is rigid enough to support any rider(s) with negligible transient effects on ride performance
4. The beam length from the center of the ride to the center of rotation of the carts (r_1) is 4.3 meters
5. r_2 is constant (the rider(s) have a negligible impact on center of mass)
6. The incline offset α_0 is 0.036 radians
7. Neglect effects from manufacturing and wear on system values
8. Given a long enough period, any realistic initial conditions will have negligible effects on the dynamics
9. Small angle approximation

1.3 Limitations

These assumptions limit this analysis method by neglecting the effect the rider(s), external forces, and transient structural behaviors have on the system's dynamics. This is important as the ride's dynamics are chaotic in nature meaning that small changes in inputs can potentially have significant impacts on the system's behavior over time. Thus it is important to note that experimental results that include the neglected factors will show behavior that deviates from the model. Additionally, this model might miss a more optimal design that comes from including these factors. Finally, by assuming constant r_2, the analysis is limited to cases where the rider(s) are still, have the same mass, and sit in the same place (which is rarely the case).

2 Optimization Method and Limitations

The optimization method chosen for this project was surrogate model based optimization. This is due to the high level of noise in the objective function as fmincon() is highly sensitive to sharp peaks and thus struggles with chaotic functions. The surrogate model was chosen to have a fifth-order isometric Matern covariance function (because it handles noisy functions with nonsimilar length scales better than other options like a Gausian radial basis function) with a Gausian likelihood function. To build the model, the Latin-Hypercube sampling method was employed to get the value of the objective function at 200 points distributed across the bounded design space. From these samples and an initial guess for the hyperparameters (see section 4), the likelihood function was maximized to optimize the hyperparameters of the surrogate model and the surrogate model was minimized using fmincon().

The SQP (Sequential Quadratic Programming) Algorithm was chosen for fmincon() (Matlab's nonlinear optimization problem solver [2]) to solve the minimization problem. The SQP algorithm uses a quasi-Newton update method to approximate the Hessian, which is used to solve the quadratic subproblems with the constraints and model linearized around the current state. Then the algorithm does a line search to determine the step size in the direction of the solution for the generalized model. The algorithm iterates through the updated states until a local minimum of the generalized model is found. To perform this analysis, the following assumptions were made:

1. Constraints are continuously twice differentiable
2. The objective function is continuously twice differentiable
3. There is a 'good' initial guess
4. There is a 'good' surrogate model

To minimize the error due to the fourth assumption, an update step was used. After fmincon() converges to a local minimum, the objective function was sampled at that location. If the error between the model and the objective function was greater than $5 * 10^{-2}$ (which was chosen by incrementally decreasing the tolerance until the condition was no longer met in 500 iterations) the hyperparameter maximization and fmincon() were rerun with the new sample added to the model.

The methodology and assumptions impose the following limitations on this analysis method:

1. Due to the derivative assumptions, smooth surrogate model functions are required
2. Due to ending conditions, the surrogate model must closely follow the trends of the objective function without violating the smoothness condition
3. Due to the gradient-driven update method, the solution may not recover from a step outside the constraints
    - Initial guess should start solver inside constraints (algorithm may or may not work otherwise)
    - A 'bad' step can put algorithm outside constraints
4. The algorithm won't know if the minimum it found is a local or global minimum
5. If the objective function behavior cannot be adequately captured in the generalized linear model, then fmincon() can take 'bad' steps due to deviation between actual and linearized models

3 Optimization Problem

The objective of this lab is to apply the principles learned in lecture to optimize the user experience on a tilt-a-whirl modeled within the following criteria and constraints [3]:

1. The angular velocity of the ride is to be between 3 rpm and 8 rpm ($3 \leq \frac{30\omega}{\pi} \leq 8$)
2. The radial distance between the center of mass and axis of rotation should be between 0.1 and 1.5 meters ($0.1 \leq r_2 \leq 1.5$)
3. The incline angle of the hills should be less than 0.3 rad ($0 \leq \alpha_1 \leq 0.3$)

The user experience on a tilt-a-whirl can be optimized by maximizing (or minimizing the negative of) the standard deviation of the rider's angular velocity. To analyze the problem the model's state dynamics were parameterized in terms of the design variables, solved using ode45(), and used to calculate the standard deviation of the rider's angular velocity (see section 1.1 for more details). The method chosen for the minimization problem was the SQP algorithm for fmincon() run on a surrogate model of the objective function (see section 2 for more details).

4 Results

4.1 Model Verification

The first results gathered were used to verify that the model described in section 1.1 was implemented correctly. To do this a plot of the objective function was made along the ω direction of the function with T=500 hills, α_1=0.058 rad, and r_2=0.8 m and compared with the provided plot on piazza [4]. The initial state conditions selected for use in all sections were $\phi_1 = \frac{-\pi}{2}$ and $\phi_2 = 0$ as the ride was started from rest on a hill (the initial position of the center of mass is expected to be straight downhill of the axis of rotation, which is a quarter of a rotation from 0 in the provided reference frame [1]).

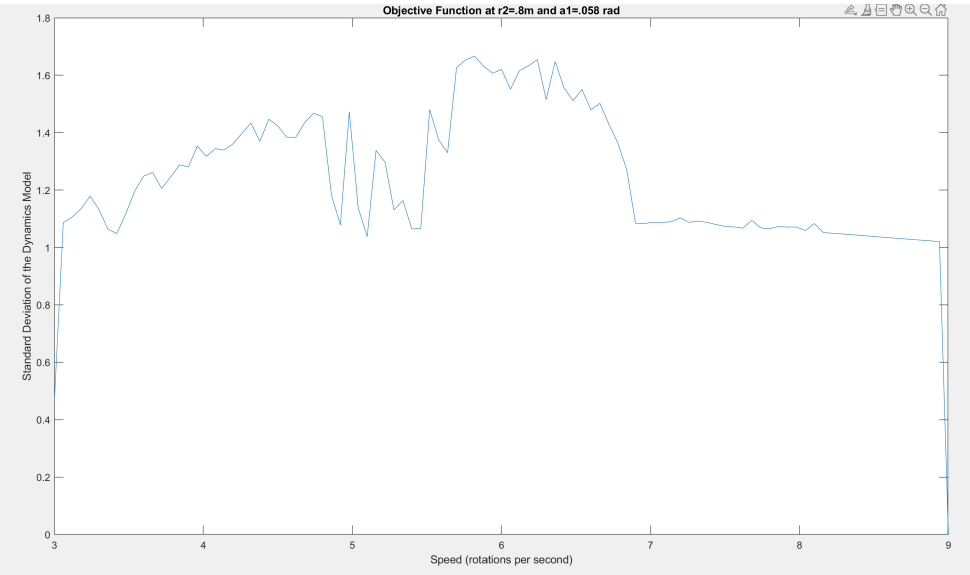

Figure 1: Generated plot of the dynamics standard deviation vs angular velocity

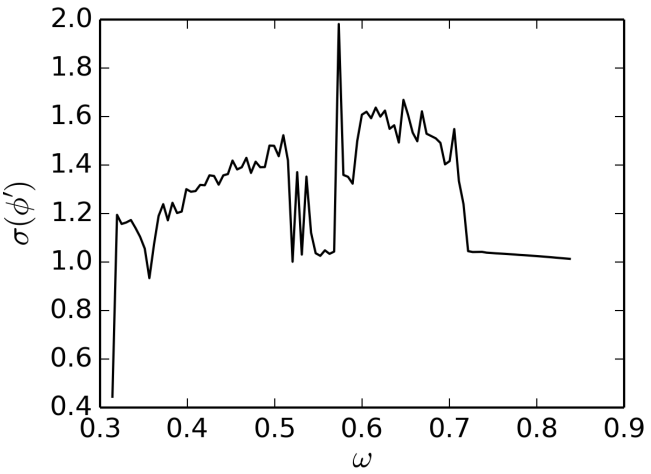

Figure 2: Provided plot of the dynamics standard deviation vs angular velocity [4]

A comparison of figures 1 and 2 shows that the objective function produces similar results to the provided plot. The broad behavior of both plots show an increase in the standard deviation of the movement to a maximum around of .6 rad/sec (6 rot/sec) before dropping and flatlining around .7 rad/sec (7 rot/sec). This behavior is also similar to the behavior described in [1]. This makes sense within the context of the problem as when ω is small the cart doesn't have enough energy to spin erratically. Additionally, when ω is too large the centripetal acceleration causes the center of mass to be flung outwards harder than the force of gravity can counteract resulting in limited eccentricity.

4.2 Convergence

The convergence plot shown in figure 3 was calculated over a range of movement cycles for the design parameters ω=6.5 rot/sec, α_1=0.058 rad, and r_2=0.8 m with the same initial states as in the previous section. As is seen in figure 3 the dynamics function roughly converges to steady state around a value of 1.55 after 700-800 hills, so T=800 hills was chosen for use in optimization.

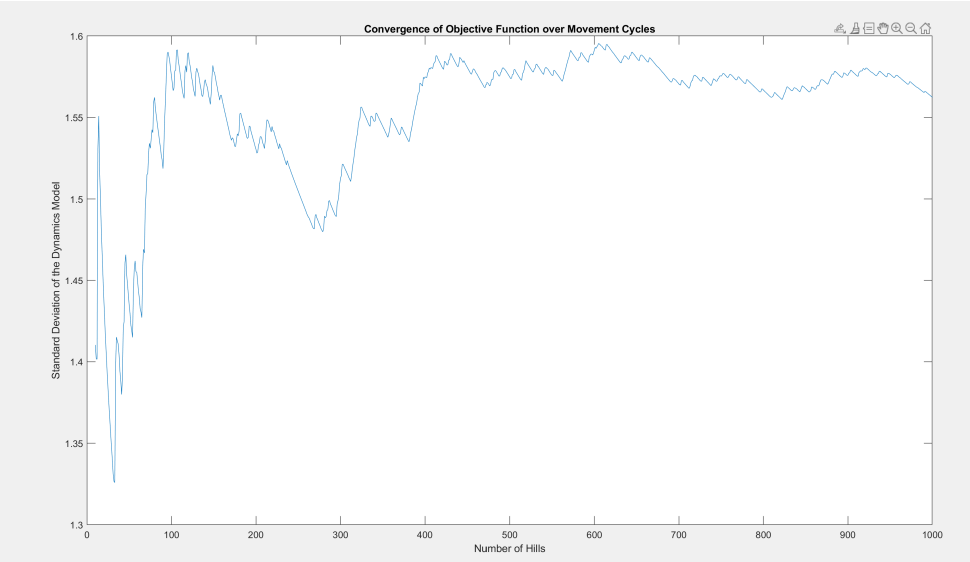

Figure 3: Plot of the standard deviation of the model over the number of hills gone over

4.3 Optimization Results

The initial hyper parameters were approximated using the results in 4.1 were chosen from the parts of figure 1 as shown in figure 4.

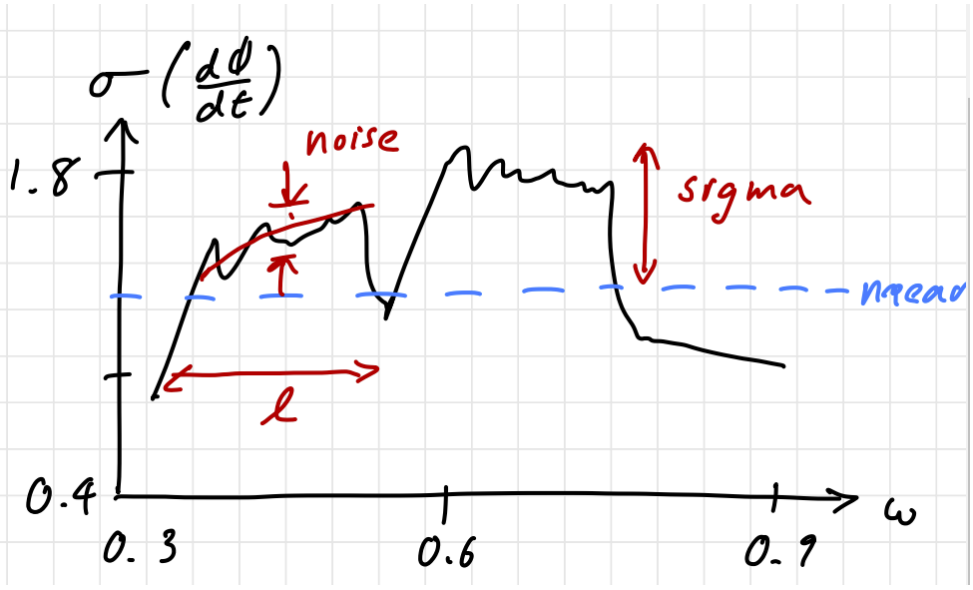

Figure 4: Provided drawing for calculating GPML hyper-parameters [5]

All parameters used in the final optimization were adjusted between optimization attempts until a consistent result was reached each time the optimization was run (in a reasonable time) and are shown in table 1.

Table 1: Design Parameters

Parameter	Value	Units
noise	.4	
L	.4	
sigma	.75	
T	800	hills
Absolute tolerance	$5 * 10^{-2}$	
Iterative tolerance	10^{-9}	
Initial ϕ_1	$\frac{-\pi}{2}$	rad
Initial ϕ_2	0	rad/s
lhssamples	200	
Maximum Optimization Iterations	200	

Slices of the objective function and surrogate models along ω are shown in figures 5 and 6 for the nominal and final designs respectively, demonstrating a reasonably good fit to the trends in the objective function. The results of the optimization are summarized in table 2 along with a comparison with the nominal values.

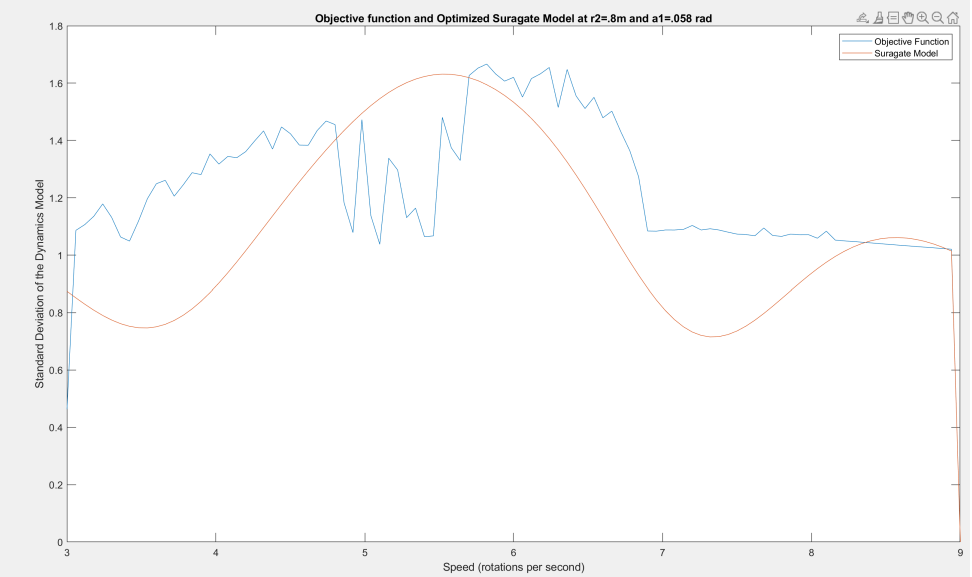

Figure 5: Generated plot of the dynamics standard deviation vs angular velocity

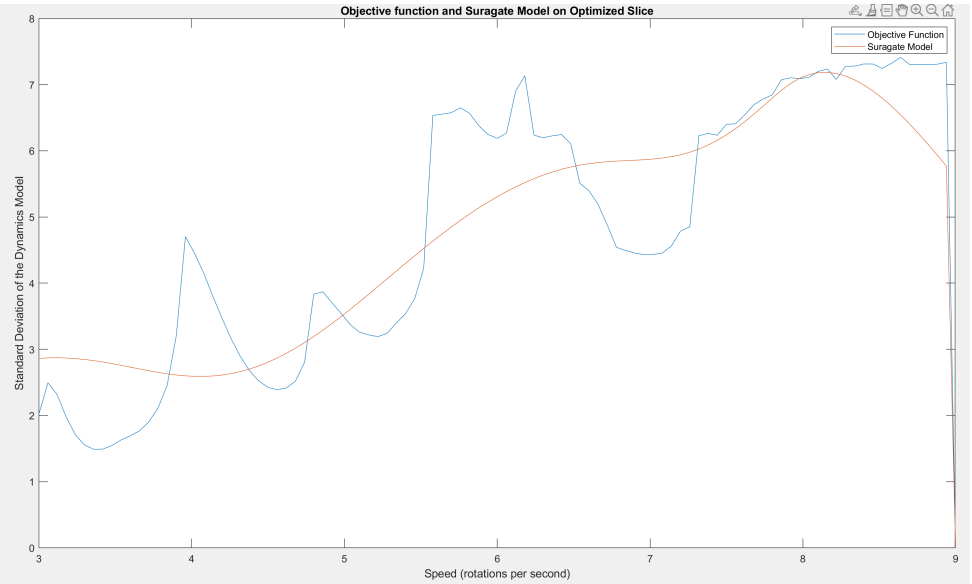

Figure 6: Generated plot of the dynamics standard deviation vs angular velocity with final design parameters

Table 2: Results

Design	ω (rot/s)	α_1 (rad)	r_2 (m)	Objective Value
Nominal	6.5	0.058	0.8	1.5668
Optimized	8	0.3	0.1	7.1184

While the optimization algorithm was running, the iteration outputs were monitered and consistently showed a feasibility score of 0 with a first-order optimality that would decrease at least six orders of magnitude each time fmincon() was run. The final result of the optimization was a 354 percent increase in the standard deviation of the angular velocity of the cart.

5 Conclusions

The implementation of the SQP optimization algorithm with the surrogate model for this lab was successful. The resulting standard deviation represents a 354 percent increase in the standard deviation of the angular velocity of the cart over the nominal parameters while remaining within the problem constraints. The resulting parameters are at the upper bounds of the angular velocity and hill size and the lower bound of the radius. These results make sense because a large hill with a small radius and high speed would cause the cart to spin erratically.

However, these results may not be representative of real implementations. This is because the model had limitations imposed on it by the simplifying assumptions. The assumption of the ride being at steady state through the convergence study was used to limit the effect of the initial states on the results. However, this means that the ride was optimized for a run time exceeding 30 minutes which is unrealistic. Additionally, the dynamics model assumed negligible effects from the riders, which is unrealistic as riders will have a noticeable affect on the location of the center of mass and will likely move during the ride.

Basic optimization algorithms, such as the ones used in this lab, can be powerful tools for understanding the complex interactions between different design variables in engineering design problems and balancing the design variables to arrive at an optimal solution. Understanding how to extend optimization algorithms to problems that are too noisy for them to efficiently solve through the use of surrogate models. Knowing how these algorithms work and the trade-offs between accuracy and run time is important when using these optimization algorithms and surrogate models together to solve problems.

References

[1] R. L. Kautz and B. M. Huggard, *Chaos at the amusement park: Dynamics of the tilt-a-whirl*, Journal, 1994.

[2] *Fmincon*, MathWorks Help Website, 2023.

[3] D. J. Hicken, *Mane 4280/6710: Numerical design optimization lab 3*, MANE 6710 Course LMS Website, October, 2024.

[4] D. J. Hicken, *Nominal design*, MANE 6710 Piazza Post, October, 2024.

[5] D. J. Hicken, *Mane 4280/6710: Numerical design optimization lab 3 notes*, MANE 6710 Course LMS Website, October, 2024.

6 Appendix

6.1 dynamics.m

```
function [phidot]=dynamics(tau,phi0,r2, alpha1,omega)
%calculate the derivative of the dynamics
% Inputs:
%   tau - the nondimensional time to evaluate the dynamics at
%   phi0 - the angular velocity and position of the car
%   r2 - the distance between the cars center of gravity and piviot point
%   alpha1 - the change in angle with respect to tau
%   omega - the angular velocity of the ride
% Outputs:
%   phidot - the angular velocity and acceleration of the car
%   ----------------------------------

%set up constants
q0=20;
r1=4.3;
alpha0=.036;
%calculate coefficients
alpha= alpha0-alpha1*cos(tau);
beta=3*alpha1*sin(tau);
gamma=sqrt(9.81/r2)/(3*omega);
e=r1/(9*r2);

%calculate dynamics
phidot=[phi0(2);-1*((e-gamma*gamma*alpha)*sin(phi0(1))+gamma*gamma*beta*
    cos(phi0(1))+gamma*phi0(2)/q0)];
```

6.2 obj.m

```
function [sigma] = obj(X, numRot)
%calculate the standard deviation of the dynamics
% Inputs:
%   X0 - (r2) the distance between the cars center of gravity and piviot
    point
%   X1 - (alpha1) the change in angle with respect to tau
%   X2 - (omega) the angular velocity of the ride
%   numRot - the number of rotations to evaluate the ride at
% Outputs:
%   sigma - the standard deviation of the dynamics
%   ----------------------------


%set ode 45 vars
%tspan from 0 to 2pi*numRot because it goes in numRot full circles
tspan=[0,numRot*2*pi];
%x0 -pi/2 because the ride at rest should point down hill, 0 bc at rest
x0=[-pi/2,0];
odefunc=@(t,y) dynamics(t,y,X(1),X(2),X(3));
%run ode45

[t,y]=ode45(odefunc,tspan,x0);
%plot(t,y(:,2))
%hold on
%plot([0,tspan(2)],[mean(y(:,2)),mean(y(:,2))])
%plot(t,y(:,1))
%hold off
%ouput standard deviation
sigma=3*X(3)*sqrt(trapz(t,((y(:,2)-trapz(t,(y(:,2)))/tspan(2)).^2))/(tspan
    (2)));
%sigma=3*X(3)*sqrt(trapz(t,((y(:,2)-trapz(t,(y(:,2)))/max(size(y))).^2))/(
    tspan(2)));

%
```

6.3 run.m

```
maxItter =200;
changeTol=10^-9;
accuracyTol=5*10^-2;

% Initialize x
%r20 = [.1,.8,1.5];
%a0=[0,.15,.3];
%w0=[3*pi/30,5*pi/30,8*pi/30];
%x=zeros([length(r20)*length(a0)*length(w0),3]);
%for i=1:length(r20)
%    for j=1:length(a0)
%        for k=1:length(w0)
%            x(length(a0)*length(w0)*(i-1)+length(w0)*(j-1)+k,:)=[r20(i),
    a0(j),w0(k)];
%        end
%    end
%end
x=lhsdesign(200,3);
x(:,1)=.1+x(:,1)*1.4;
x(:,2)=0+x(:,2)*.3;
x(:,3)=3*pi/30+x(:,3)*5*pi/30;
% where you saved the gpml directory
mydir = '../gpml-matlab-v36/';
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'])

% define the function to be sampled
fe = @(x) -1*obj(x,800);

% set the squared exponential covariance function
covfunc = {@covMaterniso, 5}; %
hyp.cov = [log(.4); log(.75)]; % first component is log(l) and second is
    log(sigma)

% set the likelihood function to Gaussian
likfunc = @likGauss;
sn = .4; %1e-16; % this is the noise level
hyp.lik = log(sn);

%set fmincon options
options = optimoptions('fmincon','Display','iter','Algorithm','sqp');
lb=[.1,0,3*pi/30];
```

```
ub=[1.5,.3,8*pi/30];

a0=[(ub(1)+lb(1)-.5)/2; (ub(2)+lb(2)+.1)/2;(ub(3)+lb(3)+.45)/2];
count=0;
% sample the function at xs
    y=zeros([length(x),1]);
    for k=1:length(x)
        y(k)=fe(x(k,:));
    end
errorobj=[];
erroritter=[];
while count<maxItter



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


    fx=@(z) gp(hyp, @infExact, [], covfunc, likfunc, x, y, z');
    %run optimization function
    a=fmincon(fx,a0,[],[],[],[],lb,ub,[],options);
    %convergend stopping conditions
    if  abs(fx(a)+fe(a))<accuracyTol
        if all(abs(fx(a)-fx(a0))<changeTol)
            break
        end
    end
    %update variables
    errorobj=[errorobj;fx(a)+fe(a)];
    erroritter=[erroritter;fx(a)-fx(a0)];
    x=[x;a'];
    y=[y;fe(a)];
    a0=a;
    count=count+1;
end
a
```

6.4 plots.m

```
    %%run fmincon first
    % where you saved the gpml directory
mydir = '../gpml-matlab-v36/';
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'])
%define actual generalized model
    fx=@(z) gp(hyp, @infExact, [], covfunc, likfunc, x, y, z);

xx=3:6/100:9;
yx=zeros(size(xx));
ys=zeros(size(xx));
rx=zeros(size(xx));
rs=zeros(size(xx));
%evaluate objective and generalized models
for i=1:100
    yx(i)=obj([.8,.058,xx(i)*pi/30],800);
    ys(i)=-1*fx([.8,.058,xx(i)*pi/30]);
    rx(i)=obj([a(1),a(2),xx(i)*pi/30],800);
    rs(i)=-1*fx([a(1),a(2),xx(i)*pi/30]);
end
%plot results
plot(xx,yx)
hold on
%plot(xx,ys)
%legend(["Objective Function", "Suragate Model"])
%title("Objective function and Optimized Suragate Model at r2=.8m and a1
   =.058 rad")
title("Objective Function at r2=.8m and a1=.058 rad")
ylabel("Standard Deviation of the Dynamics Model")
xlabel("Speed (rotations per second)")
hold off
pause
plot(xx,rx)
hold on
plot(xx,rs)
legend(["Objective Function", "Suragate Model"])
title("Objective function and Suragate Model on Optimized Slice")
ylabel("Standard Deviation of the Dynamics Model")
xlabel("Speed (rotations per second)")
hold off

%%
```

```
T=10:1000;
sigma=zeros(size(T));
for i=1:numel(T)
    sigma(i)=obj([.8,.058,6.5*pi/30],T(i));
end
plot(T,sigma)
hold on
title("Convergence of Objective Function over Movement Cycles")
ylabel("Standard Deviation of the Dynamics Model")
xlabel("Number of Hills")
hold off
```

6.5 fmincon Output

Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
0	4	-6.830077e+00	0.000e+00	1.000e+00	0.000e+00	9.095e+00
1	20	-6.830078e+00	0.000e+00	1.384e-02	2.586e-04	8.970e+00
2	24	-6.830080e+00	0.000e+00	1.000e+00	6.327e-05	1.018e-05
3	25	-6.830080e+00	0.000e+00	7.000e-01	3.886e-17	1.192e-07

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.

Figure 7: Example outpt of fmincon