P ~ Type I Extreme Value(mean=2000, cov=0.2)

E ~ Lognormal(mean=30000, cov=0.1)

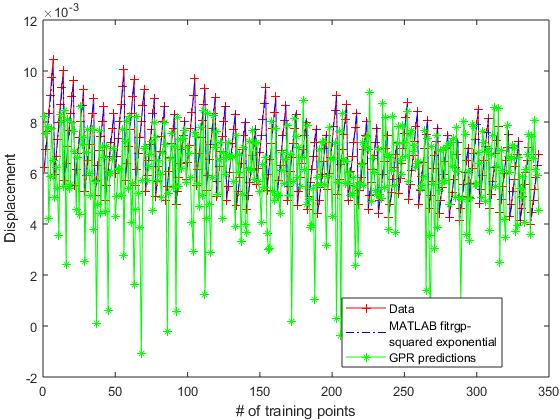
I ~ Normal(mean=10, cov=0.05)

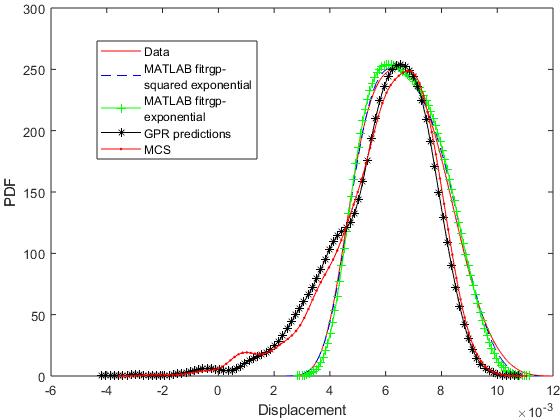
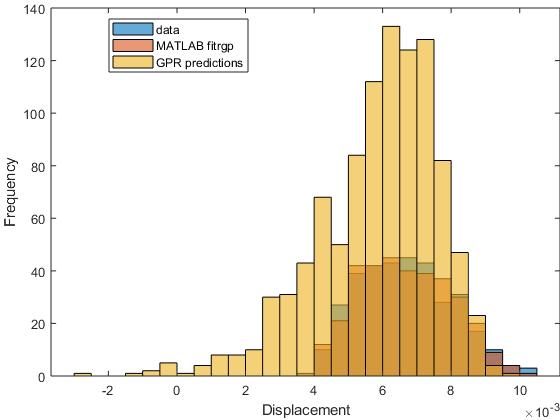
Create a surrogate model for Y = P/EI using Gaussian Process

The squared-exponential correlation function is used:

where the covariance between outputs of the Gaussian process **Z**(.) at points **a** and **b** is defined as:

Generate training points and compute output for each combination. Then, express input in terms of standard random variables and do linear regression analysis to find the coefficients.

1. Choose an initial value for
2. Assuming we have little prior knowledge about , we need to maximize for 7 training points for each dimension .
3. Compare results with MATLAB built-in function "*fitrgp*”.



clc;

clear all;close all;

tic

% P~Type I

% Type I Extreme Value Distribution:

% CDF: F\_Y = exp(-exp(-alphan\*(y-u)))

% PDF: f\_Y = alphan\*exp(-alphan\*(y-u))\*exp(-exp(-alphan\*(y-u)))

% alphan = 1/sqrt(6)\*(pi/sigmaYn)

% u = mu\_Y - 0.5772/alphan

mu\_x1 = 2000;

cov\_x1 = 0.2;

sigma\_x1 = mu\_x1\*cov\_x1;

alphan = 1/sqrt(6)\*(pi/sigma\_x1);

u = mu\_x1 - 0.5772/alphan;

F = exp(-exp(-alphan\*(mu\_x1-u)));

f = alphan\*exp(-alphan\*(mu\_x1-u))\*exp(-exp(-alphan\*(mu\_x1-u)));

sigmaN\_x1 = normpdf(norminv(F))/f;

muN\_x1 = mu\_x1 - sigmaN\_x1\*(norminv(F));

% E~Lognormal

mu\_x2 = 30000;

cov\_x2 = 0.1;

sigma\_x2 = mu\_x2\*cov\_x2;

zeta\_E = sqrt(log(1+cov\_x2^2));

lam\_E = log(mu\_x2) - 0.5\*zeta\_E^2;

lambda\_x2 = log(mu\_x2 / sqrt(1 + sigma\_x2^2/mu\_x2^2));

zeta\_x2 = log(1 + sigma\_x2^2/mu\_x2^2);

sigmaN\_x2 = zeta\_E\*mu\_x2;

muN\_x2 = mu\_x2\*(1-log(mu\_x2)+lam\_E);

% I~Normal

muN\_x3 = 10;

cov\_x3 = 0.05;

sigmaN\_x3 = muN\_x3\*cov\_x3;

% deviation from the mean

dev1 = 1.15;

dev2 = 1.25;

dev3 = 2.85;

%% Training points

n\_train = 7;

P = linspace( mu\_x1 - dev1\*sigma\_x1, mu\_x1 + 1.5\*sigma\_x1, n\_train);

E = linspace( mu\_x2 - 1.05\*sigma\_x2, mu\_x2 + 1.25\*sigma\_x2, n\_train);

I = linspace( muN\_x3 - 1.5\*sigmaN\_x3, muN\_x3 + dev3\*sigmaN\_x3, n\_train);

[PP, EE, II] = ndgrid(P,E,I);

% Input

inp = [PP(:) EE(:) II(:)];

% Output

Y = inp(:,1)./(inp(:,2).\*inp(:,3));

% Input in terms of Standard Random Variates

x1 = icdf('Normal',cdf('Extreme Value', inp(:,1), mu\_x1, sigma\_x1), 0, 1);

x2 = (log(inp(:,2))-lambda\_x2)/zeta\_x2;

x3 = ((inp(:,3)-muN\_x3)/sigmaN\_x3);

%% Linear regression analysis to find the trend

X = [ones(size(Y)) x1 x2 x3];

b = regress(Y, X);

% Residual (error)

residual = Y-X\*b;

% x1 = (x1 - min(x1))/(max(x1)-min(x1));

% x2 = (x2 - min(x2))/(max(x2)-min(x2));

% x3 = (x3 - min(x3))/(max(x3)-min(x3));

max(x1)-min(x1)

max(x2)-min(x2)

max(x3)-min(x3)

xi = [x1 x2 x3];

% constant basis (default)

gprMdl = fitrgp(xi,Y,'KernelFunction','ardsquaredexponential');

gprMd2 = fitrgp(xi,Y,'KernelFunction','ardexponential');

%

% sigma0 = 0.2;

% kparams0 = [3.5, 6.2];

% gprMdl = fitrgp(X,Y,'KernelFunction','squaredexponential',...

% 'KernelParameters',kparams0,'Sigma',sigma0);

%--------------------------------------------------------------------------

%% ---------------------- Gaussian Process -------------------------

% sigma\_e = std(residual); % std of error

sigma\_y = std(Y)/sqrt(2); % default scalar initial value for the

% noise standard deviation in GP model.

sigma\_x = std(xi)'; % default length scale vector

% x\_initial = [sigma\_x; sigma\_y];

x\_initial = [5 200 .25 0.3];

f = @(x)gpobjective(x,xi,Y);

[xOpt, fval] = fmincon(f,x\_initial);

x=xOpt

k\_tt = eye(length(Y));

k\_tp = zeros(1,length(Y));

for i=1:length(Y)

for j=1:length(Y)

k\_tt(i,j)= (-0.5\*( ((xi(i,1)-xi(j,1))/x(1))^2 + ...

((xi(i,2)-xi(j,2))/x(2))^2 + ((xi(i,3)-xi(j,3))/x(3))^2));

end

end

k\_tt = x(4)^2\*exp(k\_tt);

% MCS on the original model

% -------------------------------------------------------------------------

nsamples = 343;

% Uniform numbers

r = rand(nsamples,3);

p1 = icdf('Extreme Value', r(:,1), mu\_x1, sigma\_x1);

e2 = icdf('Lognormal', r(:,2), lambda\_x2, zeta\_x2);

i3 = icdf('Normal', r(:,3), muN\_x3, sigmaN\_x3);

MCS\_disp = p1./(e2.\*i3);

% -------------------------------------------------------------------------

% End of MCS

% N\_simulations = n\_train^3;

y\_star = zeros(nsamples,1);

for i=1:nsamples

pred\_pt=randn(1,3);

for j=1:length(Y)

k\_tp(1,j)= (-0.5\*( ((pred\_pt(1)-xi(j,1))/x(1))^2 + ...

((pred\_pt(2)-xi(j,2))/x(2))^2 + ((pred\_pt(3)-xi(j,3))/x(3))^2));

end

k\_tp = x(4)^2\*exp(k\_tp);

% k\_pp=1;

% pred\_pt=[1 pred\_pt];

% y\_star(simulation,1)=k\_tp\*(k\_tt\residual)+pred\_pt\*b;

y\_star(i,1)=k\_tp\*(k\_tt\Y);

end

% End of GP model

%--------------------------------------------------------------------------

[ypred1, ysd] = resubPredict(gprMdl);

ypred2 = resubPredict(gprMd2);

figure();

histogram(Y)

hold on

histogram(ypred1)

hold on

histogram(y\_star)

xlabel('Displacement');

ylabel('Frequency');

legend({'data','MATLAB fitrgp','GPR predictions'},'Location','Best');

figure();

[fy,xy]=ksdensity(Y);

plot(xy,fy,'-r');

hold on

[f1,x1]=ksdensity(ypred1);

plot(x1,f1,'--b');

hold on

[f2,x2]=ksdensity(ypred2);

plot(x2,f2,'-+g');

hold on

[fg,xg]=ksdensity(y\_star);

plot(xg,fg,'-\*k');

hold on

% MCS pdf

[fm,xm] = ksdensity(MCS\_disp);

plot(xm,fm,'.-r')

xlabel('Displacement');

ylabel('PDF');

legend({'Data',['MATLAB fitrgp-' newline 'squared exponential'],...

['MATLAB fitrgp-' newline 'exponential'],'GPR predictions','MCS'},'Location','Best');

figure()

plot(Y,'-+r');

hold on

plot(ypred1,'-.b');

hold on

plot(y\_star,'-\*g');

ylabel('Displacement');

xlabel('# of training points');

legend({'Data',['MATLAB fitrgp-' newline 'squared exponential'],...

'GPR predictions'},'Location','Best');

toc