**Bayesian Inference**

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

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

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

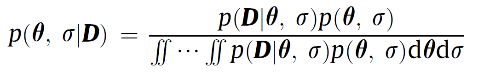
Create a surrogate model for Y = (L^3)P/3EI => y = c.x1/(x2.x3) using Gaussian Process

The actual system response is



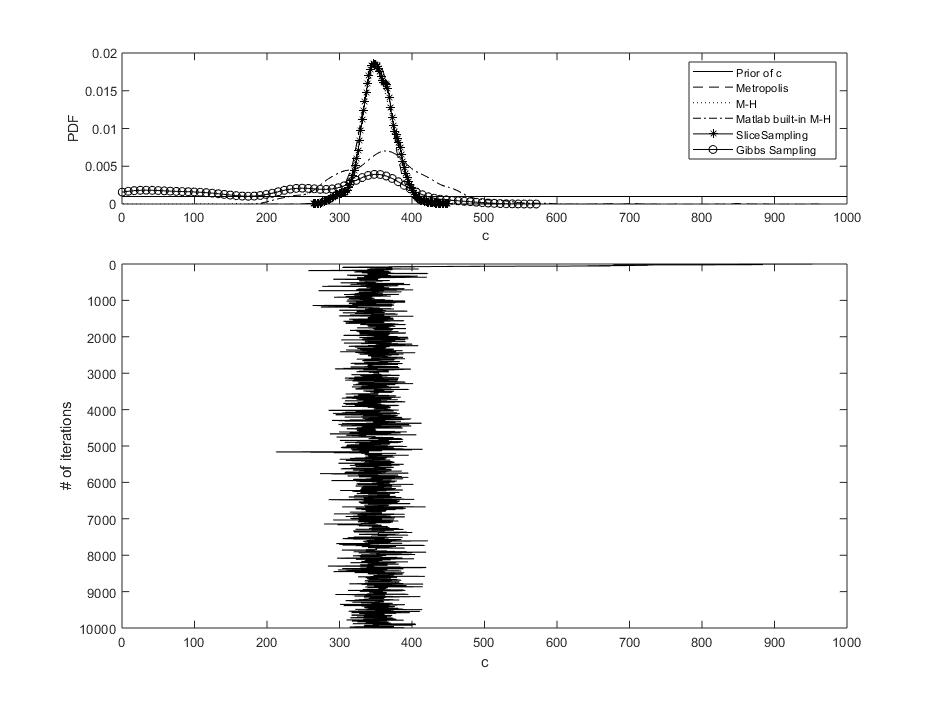
The errors are assumed to be normally distributed with zero mean and standard deviation **.**

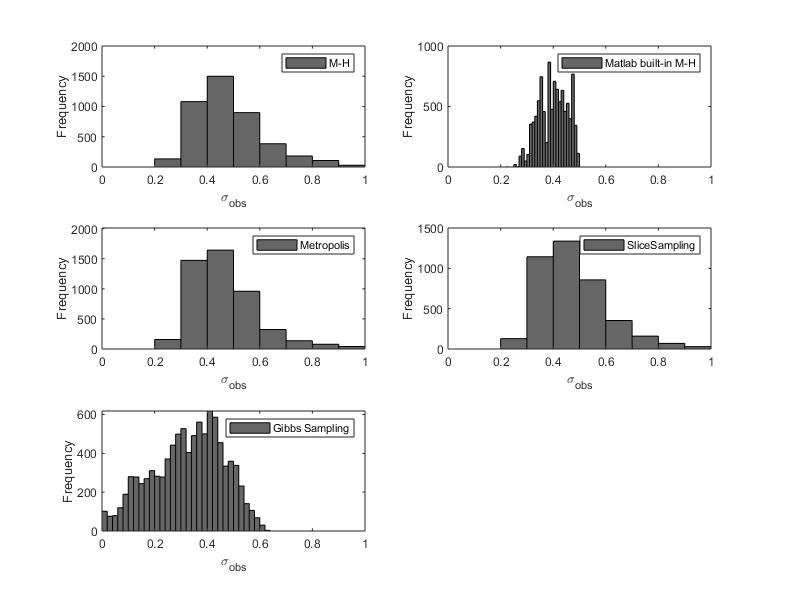


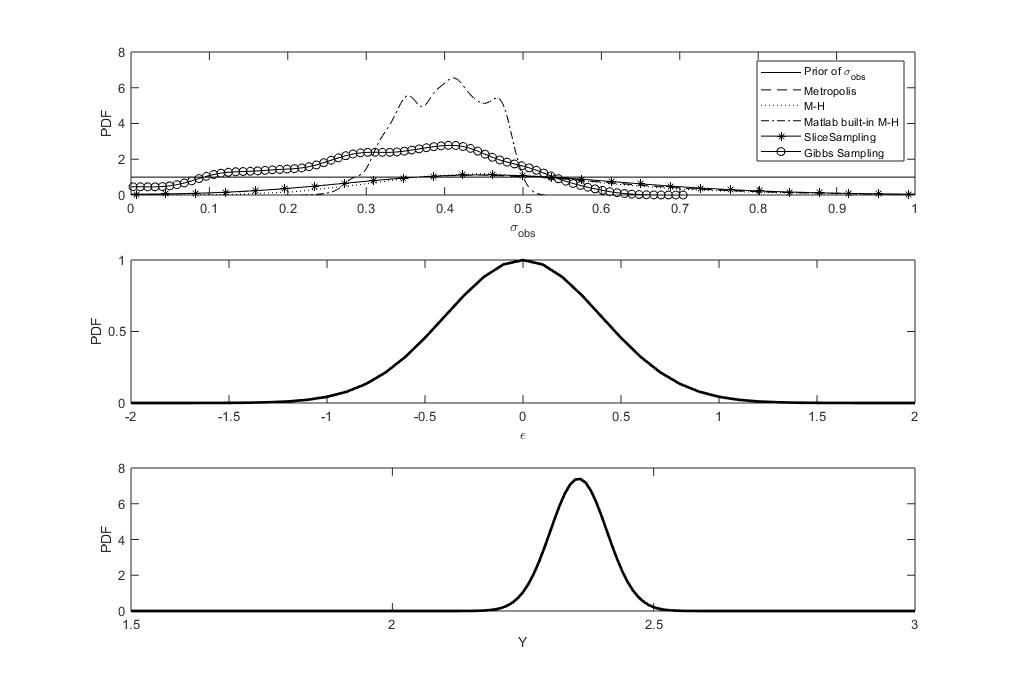
The posterior distribution of the uncertain parameters can be inferred using Bayesian approach

**Case I**

* **, where x1, x2, x3 are deterministic**

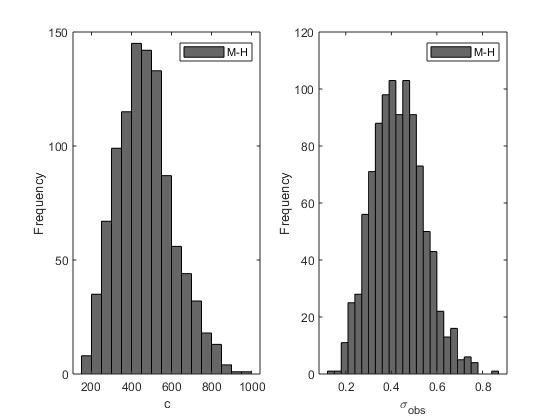






**Case II**

* **, where x1, x2, x3 are stochastic**



% clc;

% clear;

close all

%% Assume uncorrelated inputs

% Grid Samples

% P~Type I

mu\_x1 = 2000;

cov\_x1 = 0.2;

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

% E~Lognormal

mu\_x2 = 30000;

cov\_x2 = 0.1;

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

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

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

% I~Normal

muN\_x3 = 10;

cov\_x3 = 0.05;

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

dev = 3;

N\_train=5;

P = linspace( mu\_x1 - dev\*sigma\_x1, mu\_x1 + dev\*sigma\_x1, N\_train);

E = linspace( mu\_x2 - dev\*sigma\_x2, mu\_x2 + dev\*sigma\_x2, N\_train);

I = linspace( muN\_x3 - dev\*sigmaN\_x3, muN\_x3 + dev\*sigmaN\_x3, N\_train);

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

% Input

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

% Output

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

%

%% MCS

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

nsamples = 10;

% Uniform numbers

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

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

r = rand(nsamples,3);

x1 = icdf('Extreme Value', r(:,1), mu\_x1, sigma\_x1)';

x2 = icdf('Lognormal', r(:,2), muN\_x2, sigmaN\_x2)';

x3 = icdf('Normal', r(:,3), muN\_x3, sigmaN\_x3)';

Y =@(c) c.\*mu\_x1./(mu\_x2.\*muN\_x3);

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

P = mu\_x1; E = mu\_x2; I = muN\_x3;

L = 10;

% beam deflection Euler-Bernoulli

mu\_y = P\*L^3/(3\*E\*I);

sigma\_y = 0.33;

% Generate 10 samples of Y~N(mu\_y,sigma\_y)

N = 10;

yobs = normrnd(mu\_y,sigma\_y,1,N);

%% Case I

% x1, x2, x3 are deterministic at their mean values

%% Initialize the Metropolis sampler

T = 10000; % Set the maximum number of iterations

sigma\_c = 35; % Set standard deviation of normal proposal density

sigma\_eps = 0.3; % Set standard deviation of normal proposal density

cmin = 0; cmax = L^3; % define a range for starting values

c = zeros( 1 , T ); % Init storage space for our samples

sigma\_e = zeros( 1 , T ); % Init storage space for our samples

seed=1; rand( 'state' , seed ); randn('state',seed ); % set the random seed

% Prior distribution for c = L^3/3

% Assume uniform (non-informative prior)

c(1) = unifrnd( cmin , cmax ); % Generate start value

pri\_c = 1/(cmax-cmin);

smin = 0; smax = 1; % define a range for starting values

sigma\_e(1) = unifrnd( smin , smax ); % Generate start value

% err =@(s) normrnd(0,s,1,N);

pri\_s = 1/(smax-smin);

% likelihood dist.

L =@(c,s) 1/sqrt(2\*pi\*s^2)^N \* exp(-0.5\*((yobs-Y(c))\*(yobs-Y(c))')/s^2 );

% posterior dist.

post =@(c,s) L(c,s)\*pri\_c\*pri\_s;

%% Start sampling Metropolis

t = 1;

while t < T % Iterate until we have T samples

t = t + 1;

% Propose a new value for theta using a normal proposal density

c\_star = normrnd( c(t-1) , sigma\_c );

sigma\_star = normrnd( sigma\_e(t-1) , sigma\_eps );

% Calculate the acceptance ratio

alpha = min( [ 1 post( c\_star, sigma\_star ) / post( c(t-1),sigma\_e(t-1) ) ] );

% Draw a uniform deviate from [ 0 1 ]

u = rand;

% Do we accept this proposal?

if u < alpha

c(t) = c\_star; % If so, proposal becomes new state

sigma\_e(t) = sigma\_star; % If so, proposal becomes new state

else

c(t) = c(t-1); % If not, copy old state

sigma\_e(t) = sigma\_e(t-1); % If so, proposal becomes new state

end

end

%% Start sampling M-H

init=[0 0.2];

% posterior dist.

logpost =@(c,s) log(L(c,s))+log(pri\_c)+log(pri\_s);

D = numel(init);

samples = zeros(D, T);

state = init;

Lp\_state = logpost(state(1),state(2));

for ss = 1:T

% Propose

prop = state + [sigma\_c sigma\_eps].\*randn(size(state));

Lp\_prop = logpost(prop(1),prop(2));

if log(rand) < (Lp\_prop - Lp\_state)

% Accept

state = prop;

Lp\_state = Lp\_prop;

end

samples(:, ss) = state(:);

end

%% uniform prior for c

X\_c = linspace( cmin , cmax , 100 );

y\_c = unifpdf(X\_c,cmin,cmax);

% uniform prior for sigma

X\_s = linspace( smin , smax , 100 );

y\_s = unifpdf(X\_s,smin,smax);

%% DO SLICE SAMPLING

% likelihood dist.

L =@(th) 1/sqrt(2\*pi\*th(2)^2)^N \* exp(-0.5\*((yobs-Y(th(1)))\*(yobs-Y(th(1)))')/th(2)^2 );

fun=@(th,i) 1/sqrt(2\*pi\*th(2)^2) \* exp(-0.5\*((yobs(i)-Y(th(1)))^2)/th(2)^2 );

f = @(i,fun) @(th) fun(th, i); %anonymous function whose output is an anonymous function

Likelihood= 1;

for i=1:10

l1=Likelihood;

Likelihood =f(i, fun);

LL = @(th) l1(th).\*Likelihood(th);

end

post =@(th) LL(th)\*pri\_c\*pri\_s;

logpost =@(th) log(L(th))+log(pri\_c)+log(pri\_s);

initial\_samples = [300,0.5];

nSamples = 1e4;

trace = slicesample(initial\_samples, 8000,'logpdf',logpost,'thin',5,...

'burnin', 2000);

mean(trace(:,2))

%% DO M-H

% pri\_c =@(c,s) 1./(xPrior(:,2)-xPrior(:,1))';

% L =@(c,s) 1/sqrt(2\*pi\*s)^N \* exp(-0.5\*((yobs-Y(c))\*(yobs-Y(c))')/s^2 );

% posterior dist.

proprnd = @(th) [200+(1000 - 200)\*rand, 0+(0.6-0.1)\*rand]; % proposal random sampler

% logpost =@(th) log(L(th))+log(pri\_c)+log(pri\_s);

fun=@(th,i) 1/sqrt(2\*pi\*th(2)^2) \* exp(-0.5\*((yobs(i)-Y(th(1)))^2)/th(2)^2 );

f = @(i,fun) @(th) fun(th, i); %anonymous function whose output is an anonymous function

Likelihood= 1;

for i=1:10

l1=Likelihood;

Likelihood =f(i, fun);

LL = @(th) l1(th).\*Likelihood(th);

end

post =@(th) L(th)\*pri\_c\*pri\_s;

% logpost =@(th) log(LL(th))+log(pri\_c)+log(pri\_s);

% posterior dist.

logpost =@(th) log(L(th))+log(pri\_c)+log(pri\_s);

unsymPropPdf=@(x,y) mvnpdf([x(1), x(2)],[y(1),y(2)],[0.01\*y(1)^2 0;

0 0.16\*y(2)^2]);

tic

trace\_mh = mhsample([300 0.1], 1e4,'logpdf',logpost,'proprnd',proprnd, ...

'proppdf',unsymPropPdf);

toc

post =@(th) LL(th)\*pri\_c\*pri\_s;

%% Gibbs Sampling

scale = [10 .01];

v\_th=Gibbs(post,[30 0.4],T,scale);

%

%% Display ksdensity of c

figure( 1 ); clf;

subplot( 3,1,1 );

[f1,c1]=ksdensity(c);

[f2,c2]=ksdensity(samples(1,:));

[f3,c3]=ksdensity(trace\_mh(:,1));

[f4,c4]=ksdensity(trace(:,1));

[f5,c5]=ksdensity(v\_th(:,1));

% set(groot,'DefaultAxesColorOrder',[0 0 0],...

% 'DefaultAxesLineStyleOrder','-|--|:|-.|-\*|o')

set(groot,'DefaultAxesColorOrder',[0 0 0],...

'DefaultAxesLineStyleOrder','-|--|:|-.|-\*|-o|-.\*')

plot(X\_c,y\_c);hold on;

plot(c1,f1);hold on;

plot(c2,f2);hold on;

plot(c3,f3);hold on;

plot(c4,f4);hold on;

plot(c5,f5);

xlim( [ cmin cmax ] );

xlabel( 'c' ); ylabel( 'PDF' );

legend('Prior of c','Metropolis','M-H', 'Matlab built-in M-H',...

'SliceSampling','Gibbs Sampling')

%% Display history of our samples

subplot( 3,1,2:3 );

stairs( c , 1:T , 'k-' );

ylabel( '# of iterations' ); xlabel( 'c' );

set( gca , 'YDir' , 'reverse' );

xlim( [ cmin cmax ] );

%% Display ksdensity of sigma

figure( 2 ); clf;

subplot( 3,1,1 );

plot(X\_s,y\_s);hold on;

ksdensity(sigma\_e);hold on;

ksdensity(samples(2,:));hold on;

ksdensity(trace\_mh(:,2));hold on;

ksdensity(trace(:,2));hold on;

ksdensity(v\_th(:,2));

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'PDF' );

legend('Prior of \sigma\_{obs}','Metropolis','M-H','Matlab built-in M-H',...

'SliceSampling','Gibbs Sampling')

subplot( 3,1,2 );

mu = 0;

sigma = mean(trace\_mh(:,2));

pd = makedist('Normal','mu',mu,'sigma',sigma);

x =-2:0.1:2;

y = pdf(pd,x);

plot(x,y,'LineWidth',2);

xlabel( '\epsilon' ); ylabel( 'PDF' );

subplot( 3,1,3 );

mu = mean(trace\_mh(:,1))\*P/E/I;

sigma = std(trace\_mh(:,2));

pd = makedist('Normal','mu',mu,'sigma',sigma);

x =1.5:0.01:3;

y = pdf(pd,x);

plot(x,y,'LineWidth',2);

xlabel( 'Y' ); ylabel( 'PDF' );

figure()

subplot(3,2,3)

histogram(sigma\_e)

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('Metropolis')

subplot(3,2,1)

histogram(samples(2,:));

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('M-H')

subplot(3,2,2)

histogram(trace\_mh(:,2));

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('Matlab built-in M-H')

subplot(3,2,4)

histogram(trace(:,2));

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('SliceSampling')

subplot(3,2,5)

histogram(v\_th(:,2));

xlim( [ smin 1 ] );

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('Gibbs Sampling')

% %% Display history of our samples

% subplot( 3,1,2:3 );

% stairs( sigma\_e , 1:T , 'k-' );

% ylabel( '# of iterations' ); xlabel( '\sigma\_{obs}' );

% set( gca , 'YDir' , 'reverse' );

% % xlim( [ smin 1 ] );

%% Case II

% x1, x2, x3 have given distributions

% To calculate likelihood assume error\_obs~N(0,sigma\_obs)

% We have 2 calibration quantity: c and error\_obs

% Generate 10 samples of Y~N(mu\_y,sigma\_y)

N = 10;

yobs = normrnd(2.2,0.33,1,N);

% log posterior

logpost =@(X) ProposedPosterior(X,yobs,0,1000,0,1);

% proposed random number generator

proprnd = @(X) mvnrnd([500,0.4],[150^2 0;0 0.12^2],1);

% proposed unsymmetric pdf

unsymmPropPdf=@(X,Y) mvnpdf([X(1), X(2)],[Y(1),Y(2)],[0.09\*Y(1)^2 0;

0 0.09\*Y(2)^2]);

% tic

% trace\_mm = mhsample([500 0.4], 1000,'logpdf',logpost,'proprnd',proprnd, ...

% 'symmetric',true,'thin',5,'burnin', 1000);

% toc

%

% tic

% trace\_mh = mhsample([500 0.4], 1e3,'logpdf',logpost,'proprnd',proprnd, ...

% 'proppdf',unsymmPropPdf,'thin',5,'burnin', 1000);

% toc

%

% figure()

% subplot(1,2,1)

% histogram(trace\_mm(:,1))

% xlabel( 'c' ); ylabel( 'Frequency' );

% legend('Metropolis')

% subplot(1,2,2)

% histogram(trace\_mm(:,2))

% xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

% legend('Metropolis')

%

% figure()

% subplot(1,2,1)

% histogram(trace\_mh(:,1))

% xlabel( 'c' ); ylabel( 'Frequency' );

% legend('M-H')

% subplot(1,2,2)

% histogram(trace\_mh(:,2))

% xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

% legend('M-H')

tic

trace = slicesample([500 0.4], 100,'logpdf',logpost,'thin',5,...

'burnin', 2000,'width',[100 10]);

mean(trace(:,2))

toc

figure()

subplot(1,2,1)

histogram(trace(:,1))

xlabel( 'c' ); ylabel( 'Frequency' );

legend('SliceSampling')

subplot(1,2,2)

histogram(trace(:,2))

xlabel( '\sigma\_{obs}' ); ylabel( 'Frequency' );

legend('SliceSampling')

function Y = ProposedPosterior(X, y\_obs, cmin, cmax, sigma\_min, sigma\_max)

% c = X(1), sigma\_obs = X(2)

N = 500; % # of MCS for X

% Uniform numbers (LHS)

r = [lhsdesign(N,3) rand(N,1)];

u = 0.5772;

sigmaBar = sqrt(6)\*(2000\*0.2)/pi;

muBar=2000-u\*sigmaBar;

muN\_x2 = log(30000 / sqrt(1 + 3000^2/30000^2));

sigmaN\_x2 = log(1 + 3000^2/30000^2);

%% MCS

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

x1 = icdf('GeneralizedExtreme Value', r(:,1), sigmaBar, muBar);

x2 = icdf('Lognormal', r(:,2), muN\_x2, sigmaN\_x2);

x3 = icdf('Normal', r(:,3), 10, 0.5);

% Y\_obs = Y\_m - eps\_obs

eps\_obs = icdf('Normal', r(:,4), 0, abs( X(2) ));

Y\_m = X(1)\*x1./(x2.\*x3);

Y\_obs = Y\_m - eps\_obs;

% find pdf values of corresponding y\_obs

[fobs, yobs] = ksdensity(Y\_obs);

len = length(y\_obs);

L=0;

for i=1:len

[~,idx] = min(abs(yobs-y\_obs(i)));

pdf\_value = fobs(idx);

L = L + log(pdf\_value);

end

Y = L + log(pdf('Uniform',X(1),cmin,cmax))+log(pdf('Uniform',X(2),sigma\_min,sigma\_max));

end