Multipoints-SimplySupportedBeam-CustomerLikelihood-Error Surrogate

This example is with known exact solution. Compared with analytical solution, this example hopes to provide surrogate model to create

customer defined loglikelihood to do Bayesian inference

1 - INITIALIZE UQLAB

```
clearvars
rng(100,'twister')
uqlab
```

```
Copyright 2013-2022, Stefano Marelli and Bruno Sudret, all rights reserved.
This is UQLab, version 2.0
UQLab is distributed under the BSD 3-clause open source license available at:
C:\NY2023\D_document\UQLab_Rel2.0.0\LICENSE.

To request special permissions, please contact:
  - Stefano Marelli (marelli@ibk.baug.ethz.ch).

Useful commands to get started with UQLab:
uqlab -doc            - Access the available documentation
uqlab -help           - Additional help on how to get started with UQLab
uq_citation help    - Information on how to cite UQLab in publications
uqlab -license        - Display UQLab license information
```

2 - Create Error surrogate model

90% data for model training and 1**0%** data for cross validation

Read measurements

```
Measurement = xlsread("data.xlsx",'Measurement_for_surrogate','A2:AC11');
```

Read forward model FE simulations

including 90% training data and 10% validation data

```
Forward_model = xlsread("data.xlsx",'FE_models','A2:AE291');
```

Obtain the number of experiment **expNum**, the number of output points (positon of deflection) at one experiment **Npoint**, the number of FE simulation realizaiton **Nreal**

```
expNum = size(Measurement,1);
Npoint = size(Measurement,2);
Nreal = size (Forward_model,1);
```

Root-mean-square-error to get the error

```
DiscrepancySum = [];
%Root mean square error with 90% of Nreal simulations (25 FE simulaiton, so the
first 22 FE Simulaitons are used)
```

```
for i  = 1: expNum

        discrepancy = (Measurement(i,:) - Forward_model(1:0.9*Nreal,3:31)).^2;

        DiscrepancySum = [DiscrepancySum, sum(discrepancy,2)];


end

error = sqrt(sum(DiscrepancySum,2)/expNum/Npoint);

surrogate = [Forward_model(1:0.9*Nreal,1:2),error];
```

Response surface model fitting

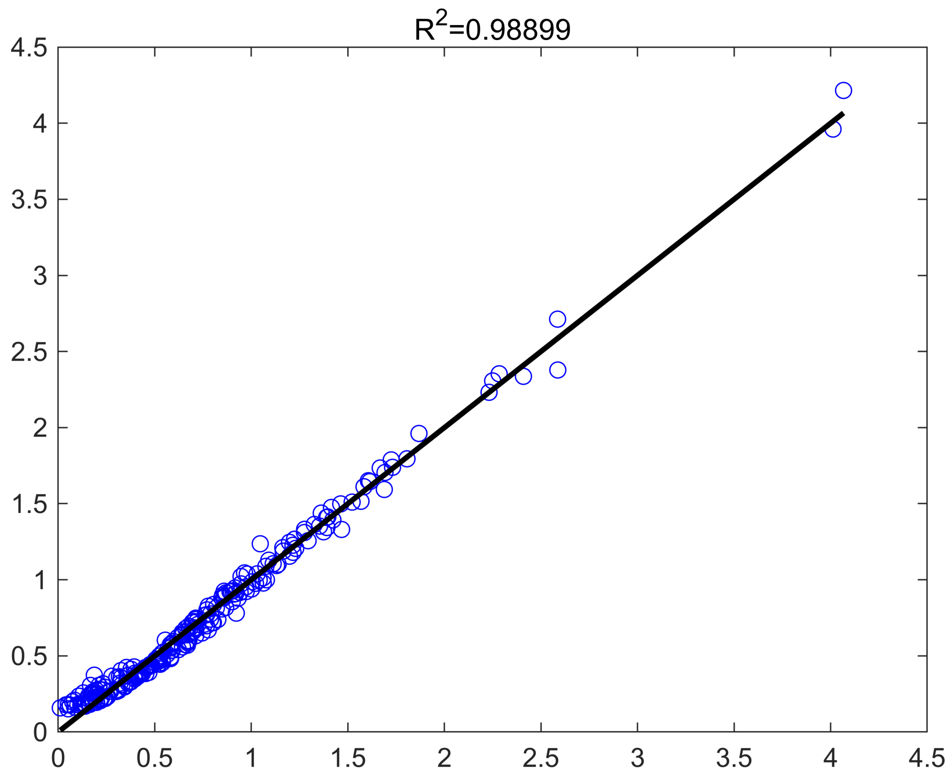
```
x(:,1) = surrogate(:,1)/10e9;% normlized the E with 10e9Pa
x(:,2) = surrogate(:,2)/5;% normalized the delta with deviatoin = 5m
y_fit = surrogate(:,3); % sum error

% response surface model-- four order polynomial equation

model_fun_RSM = @(p,x) p(1) + ...
p(2) * x(:,1) + ...
p(3) * x(:,2) + ...
p(4) * x(:,1).^2 + ...
p(5) * x(:,2).^2 + ...
p(6) * x(:,1).^3 + ...
p(7) * x(:,2).^3 + ...
p(8) * x(:,1).^4 + ...
p(9) * x(:,2).^4 + ...
p(10) * x(:,1).^3 .* x(:,2) + ...
p(11) * x(:,1).^2 .* x(:,2).^2 + ...
p(12) * x(:,1) .* x(:,2).^3 + ...
p(13) * x(:,1).^4 .* x(:,2) + ...
p(14) * x(:,1) .* x(:,2).^4 + ...
p(15) * x(:,1).^5 + ...
p(16) * x(:,2).^5 + ...
p(17) * x(:,1).^4 .* x(:,2).^2 + ...
p(18) * x(:,1).^3 .* x(:,2).^3 + ...
p(19) * x(:,1).^2 .* x(:,2).^4 + ...
p(20) * x(:,1) .* x(:,2).^5 + ...
p(21) * x(:,1).^6 + ...
p(22) * x(:,2).^6 + ...
p(23) * x(:,1).^5 .* x(:,2) + ...
p(24) * x(:,1).^4 .* x(:,2).^2 + ...
p(25) * x(:,1).^3 .* x(:,2).^3 + ...
p(26) * x(:,1).^2 .* x(:,2).^4 + ...
p(27) * x(:,1) .* x(:,2).^5 + ...
```

```
p(28) * x(:,1).^6 .* x(:,2) + ...
p(29) * x(:,1).^5 .* x(:,2).^2 + ...
p(30) * x(:,1).^4 .* x(:,2).^3 + ...
p(31) * x(:,1).^3 .* x(:,2).^4 + ...
p(32) * x(:,1).^2 .* x(:,2).^5 + ...
p(33) * x(:,1) .* x(:,2).^6;



p0 = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 1 1 ];

[p,R,J,CovB,MSE,ErrorModelInfo] = nlinfit(x,y_fit,model_fun_RSM,p0);
```

警告: 解处的 Jacobian 矩阵为病态，而且某些模型参数的估计值可能不准确(不可识别)。进行预测时要谨慎。

Surrogate mode: Predicted vs True

```
y_estimate = model_fun_RSM(p,x);
SSR = sum(R.^2);
SST = sum((y_fit - mean(y_fit)).^2);
R2 = 1 - SSR / SST;
str=['R^2=', num2str(R2)];
close all;
plot(y_fit,y_estimate,'bo');
hold on;
plot([min(y_fit),max(y_fit)],[min(y_fit),max(y_fit)],'k-','LineWidth',2);
title(str);
hold off;
```



Calculate the Leave-one-out error

```
Xval =  Forward_model((0.9*Nreal+1):Nreal,:);
E_val = Xval(:,1);
delta_val = Xval(:,2);
Deflection_val = Xval(:,3:31);
LOO_error = 0;

%%calculate  the Y value
discrepSum = [];

for i  = 1: expNum

        discrep = (Measurement(i,:) - Deflection_val).^2;

        discrepSum = [discrepSum, sum(discrep,2)];


end

Y = sqrt(sum(discrepSum,2)/expNum/Npoint);

% calculate the  Y_surrogate value and LOO_error
Y_Surrogate = 0;
Mean_Y = mean(Y);
Var_LOO = 0;
```

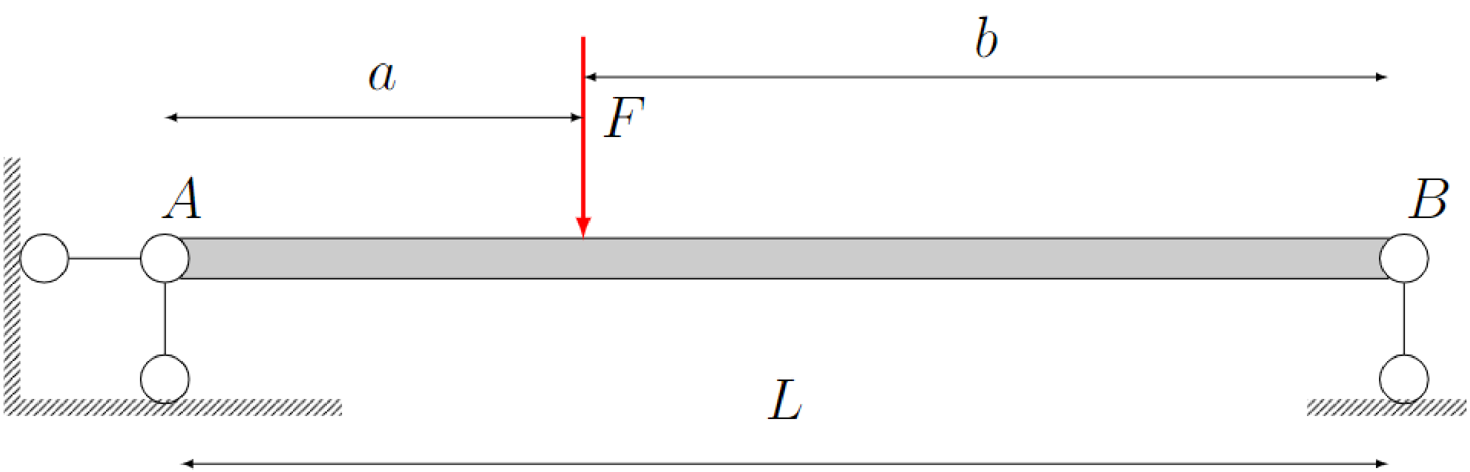
```
LOO_errorSum = 0;

for i  = 1 : size(Y,1)
    X_LOO(1) = E_val(i)/10e9;
    X_LOO(2) = delta_val(i)/5;
    Y_Surrogate = model_fun_RSM(p,X_LOO);
    LOO_errorSum  =LOO_errorSum + (Y(i) - Y_Surrogate).^2;
    Var_LOO = Var_LOO + ( Y_Surrogate - Mean_Y).^2;
end



LOO_error = LOO_errorSum / Var_LOO;


disp(strcat('Leave-One-Out Error is  ', num2str(LOO_error)));
```

```
Leave-One-Out Error is0.0071983
```

Pass the coeficients of RSM to myData.y

```
myData.y = p;
myData.Name = 'Multi points deflection along the beam';
```

3 - PRIOR DISTRIBUTION



b_b = 0.15; % beam width (m)

b_h = 0.3; % beam height (m)

a % distance from the point A (m)

b % distance from the point B (m)

L = 30; % beam length (m)

F = 43000;% Concentrated force (N)

```
%Priors on E and p
PriorOpts.Marginals(1).Name = 'E';               % Young's modulus
PriorOpts.Marginals(1).Type = 'Gaussian';
PriorOpts.Marginals(1).Moments = [25 5]*1e9;   % (N/m^2)

PriorOpts.Marginals(2).Name = 'delta';               % Concentrated load loading position
PriorOpts.Marginals(2).Type = 'Gaussian';
PriorOpts.Marginals(2).Moments = [3 5]; % (N/m)


PriorOpts.Marginals(3).Name = 'sigma2'; % variance
PriorOpts.Marginals(3).Type = 'Uniform';
sigma2 = mean(Measurement(:,:),"all");
PriorOpts.Marginals(3).Parameters = [0 sigma2^2];

myPriorDist = uq_createInput(PriorOpts);
```

4 - Define the custom-loglikelihood

Loglikelihood still follows the Gaussian discrepancy criteria

```
myLogLikeli = @(params,y) myLogLikeli2(params,y);
```

5 - Solver options

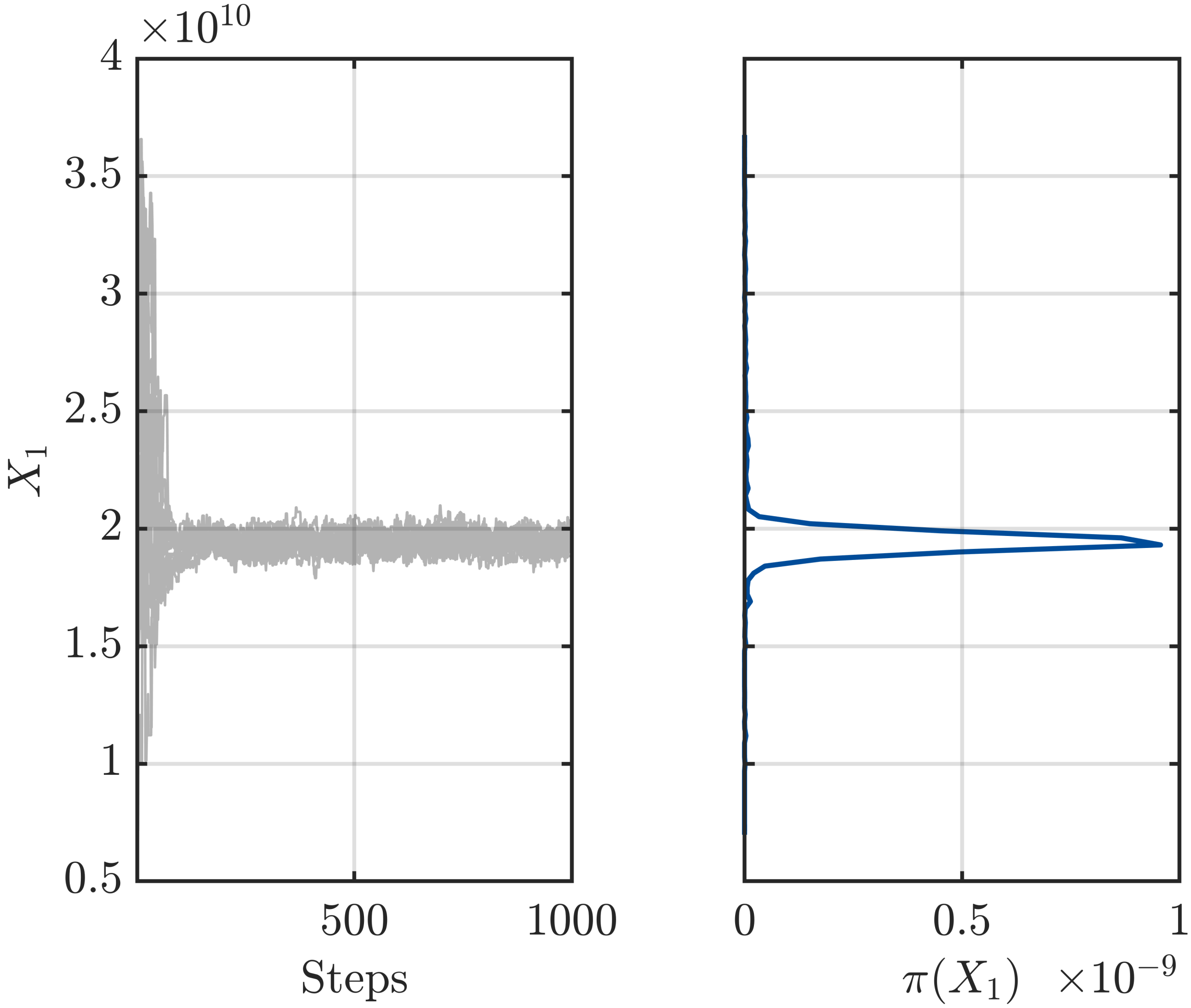
```
Solver.Type = 'MCMC';
Solver.MCMC.Visualize.Parameters = [1 2];
Solver.MCMC.Visualize.Interval = 10;
Solver.MCMC.Sampler = 'AIES';
Solver.MCMC.Steps = 1000;
Solver.MCMC.NChains = 20;
Solver.MCMC.Proposal.PriorScale = 1e-3;
```

6 - Bayes Analysis

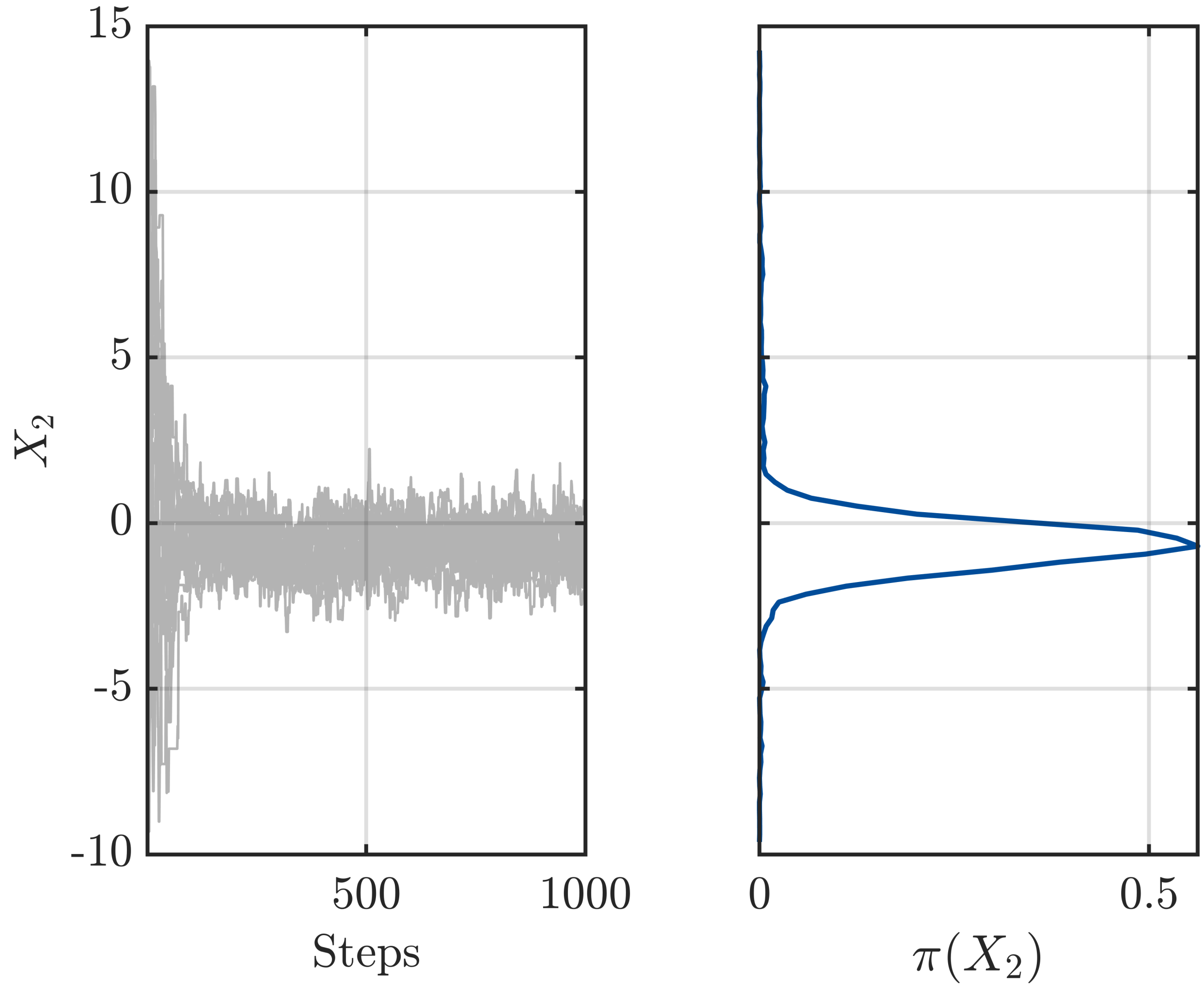
Consistent with example

```
BayesOpts.Data = myData;
BayesOpts.LogLikelihood = myLogLikeli;
BayesOpts.Type = 'inversion';
BayesOpts.Solver = Solver;
BayesAnalysis = uq_createAnalysis(BayesOpts);
```

```
Starting AIES...

|                         |   1.00%|#                               |   2.00%|#                               |   
```



$\times 10^{10}$
4
3.5
3
2.5
2
1.5
1
0.5
X_1
500
1000
Steps
0
0.5
1
$\pi(X_1)$
$\times 10^{-9}$




```
Finished AIES!
```

7 - Postprocess results

7.1 Ground Truth

Ground truth should be:

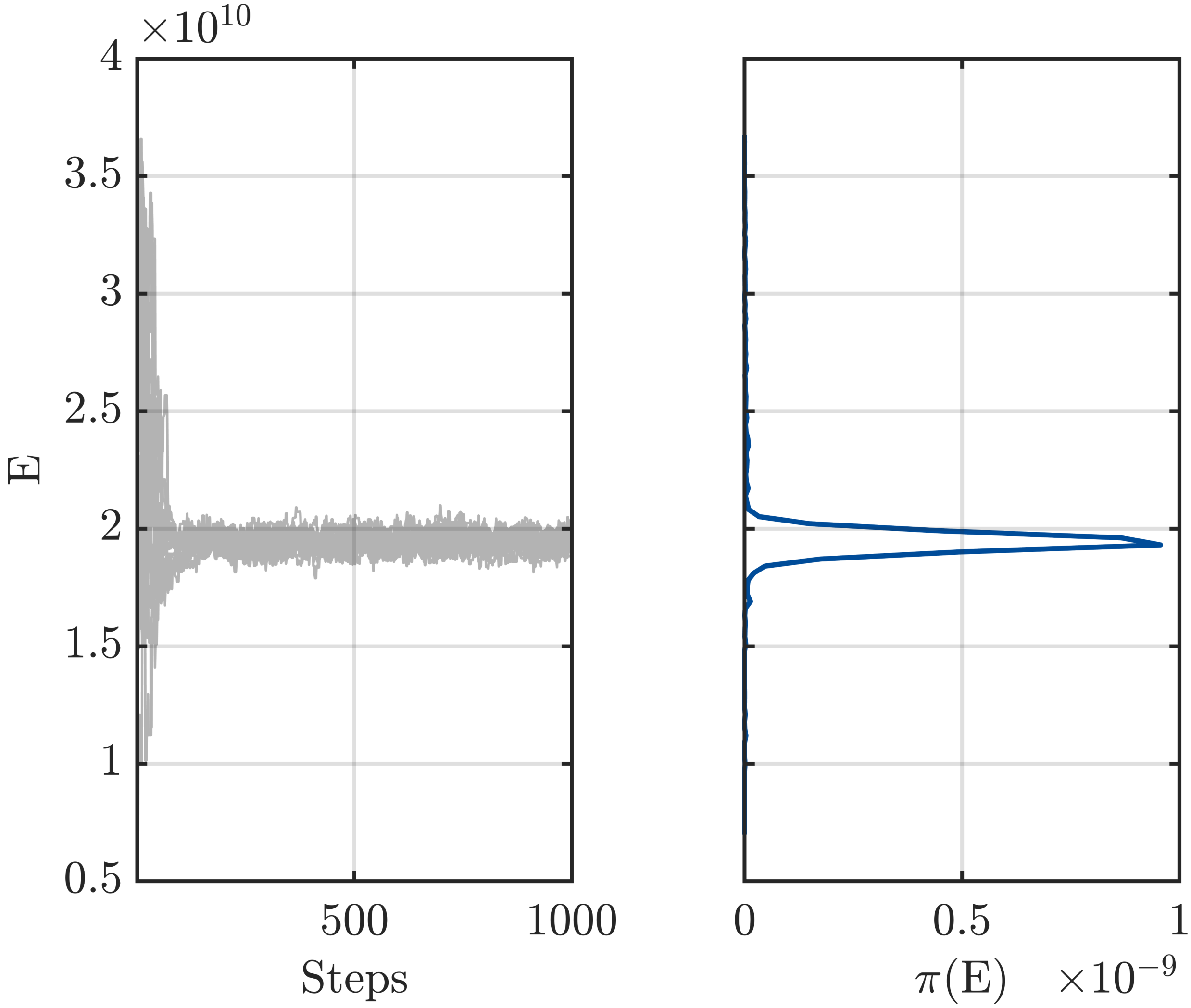
E = 19.1e9Pa; δ = -1.708m

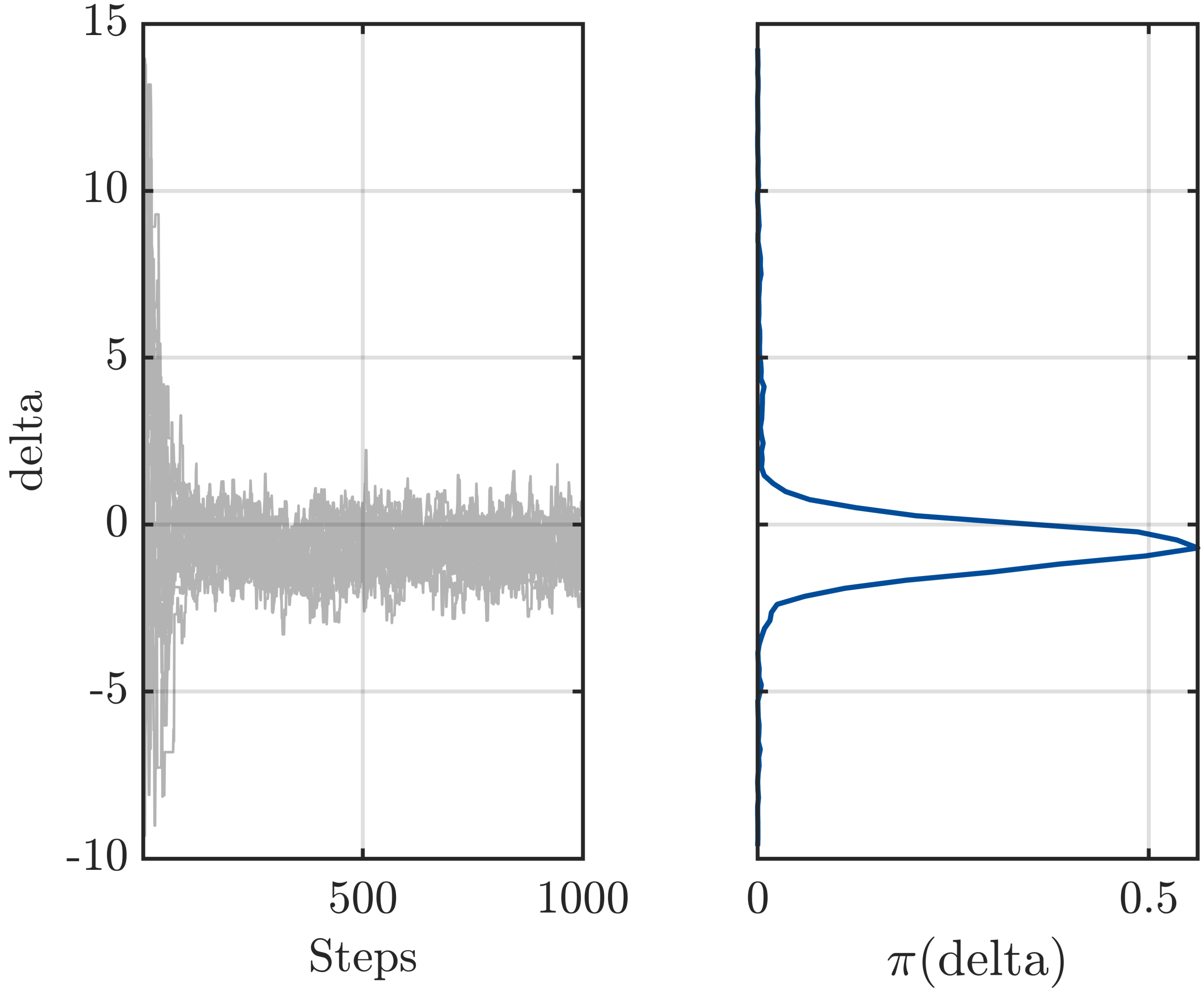
```
%display the results of bayesian inference

%set cutoff on the badchain as delta < -10m
close all;

badChainsIndex = squeeze(BayesAnalysis.Results.Sample(end,2,:) > 5);



uq_postProcessInversionMCMC(BayesAnalysis,'pointEstimate','MAP','percentiles',
[0.05,0.95],'burnin',0.7,'badChains',badChainsIndex);
```

```
uq_print(BayesAnalysis);
```

```
%----------------------- Inversion output -----------------------%
   User-specified likelihood used
%------------------- Solver
   Solution method:                               MCMC

   Algorithm:                                     AIES
   Duration (HH:MM:SS):                           00:00:41
   Number of sample points:                       2.00e+04

%------------------- Posterior Marginals
---------------------------------------------------------------
| Parameter | Mean     | Std      | (0.05-0.95) Quant.  | Type  |
---------------------------------------------------------------
| E         | 1.9e+10 | 3.8e+08 | (1.9e+10 - 2e+10)   | Model |
| delta     | -0.66   | 0.65    | (-1.8 - 0.33)       | Model |
| sigma2    | 0.00095 | 0.00028 | (0.0006 - 0.0015)   | Model |
---------------------------------------------------------------

%------------------- Point estimate
----------------------------------------
| Parameter | MAP     | Parameter Type |
----------------------------------------
| E         | 1.9e+10 | Model          |
| delta     | -0.7    | Model          |
| sigma2    | 0.00078 | Model          |
----------------------------------------

%------------------- Correlation matrix (model parameters)
---------------------------------------
|        |  E        delta    sigma2  |
---------------------------------------
| E      |  1        0.31     0.067   |
| delta  |  0.31     1        0.066   |
| sigma2 |  0.067    0.066    1       |
---------------------------------------
```

```
uq_display(BayesAnalysis, 'trace', [1 2]);
```


×10^10
4
3.5
3
2.5
2
1.5
1
0.5
E
500
1000
Steps
0
0.5
1
π(E)
×10^-9



7.2 Comparison with prior and posterior on E and export figures

Display the prior distribution and posterior distribution E

```
close all;
E = 10e9:0.05e8:21e9;
Pdf_E_Prior = normpdf(E,20e9,5e9);
Pdf_E_Posterior =
normpdf(E,BayesAnalysis.Results.PostProc.Percentiles.Mean(1),BayesAnalysis.Results.P
ostProc.Percentiles.Var(1)^0.5);
plot(E,Pdf_E_Prior,'b','LineWidth',2);
hold on;
plot(E,Pdf_E_Posterior,'c',"LineWidth",2);
xline(15e9,'--r','LineWidth',2);
xlabel("E (Pa)");
```

```
ylabel('PDF');
ax = gca;
ax.FontSize = 16;
legend('Prior','Posterior',"Ground Truth",'location','northwest');
legend('boxoff');
xlim([14e9 16e9]);
%ax =gca;
%exportgraphics(ax,'figure2.pdf','ContentType','vector');
```

7.3 Comparison with prior and posterior on δ and export figures

Display the prior distribution and posterior distribution delta

```
%Display the prior distribution and posterior distribution delta
close all;
delta = -2:0.001:3;
Pdf_delta_Prior = normpdf(delta,0,5);
Pdf_delta_Posterior =
normpdf(delta,BayesAnalysis.Results.PostProc.Percentiles.Mean(2),BayesAnalysis.Resul
ts.PostProc.Percentiles.Var(2)^0.5);
plot(delta,Pdf_delta_Prior,'b','LineWidth',2);
hold on;
plot(delta,Pdf_delta_Posterior,'c',"LineWidth",2);
xline(2,'--r','LineWidth',2);
xlabel("\delta (m)");
ylabel('PDF');
ax = gca;
ax.FontSize = 16;
legend('Prior','Posterior',"Ground Truth",'location','northwest');
legend('boxoff');
xlim([-2 2.5]);
%ax =gca;
%exportgraphics(ax,'figure3.pdf','ContentType','vector');
```

7.4 Draw the 90% and 60% error band on the beam

qqplot to see if the posterior follows Gaussian distribution

```
close all;
%loglikelihood follows Gaussian discrepancy model, thus posterior should be
%Gaussian distribution
% close all;
% qqplot(BayesAnalysis.Results.PostProc.PostSample(:,1,1));
% hold on;
% qqplot(BayesAnalysis.Results.PostProc.PostSample(:,2,1));
% qqplot(BayesAnalysis.Results.PostProc.PostSample(:,3,1));
% hold off;
```

mean value and std value of posterior E and delta

```
E_mean = BayesAnalysis.Results.PostProc.Percentiles.Mean(1);
E_std = BayesAnalysis.Results.PostProc.Percentiles.Var(1)^0.5;

delta_mean = BayesAnalysis.Results.PostProc.Percentiles.Mean(2);
delta_std = BayesAnalysis.Results.PostProc.Percentiles.Var(2)^0.5;
```

Generate 1000 sampels for the forward model

```
n = 1000;
E_sample = normrnd(E_mean,E_std,[n,1]);
delta_sample = normrnd(delta_mean,delta_std,[n,1]);
```

Set 90% and 60% cutoff on E and delta

```

%set 90% cutoff on E
CI_90 = 1.645*E_std;
E_lower_90 = E_mean - CI_90;
E_upper_90 = E_mean + CI_90;


%set 90% cutoff on delta
CI_90 = 1.645*delta_std;
delta_lower_90 = delta_mean - CI_90;
delta_upper_90 = delta_mean + CI_90;

%set 60% cutoff on E

CI_60 = 0.8416*E_std;
E_lower_60 = E_mean - CI_60;
E_upper_60 = E_mean + CI_60;


%set 60% cutoff on delta
CI_60 = 0.8416*delta_std;
delta_lower_60 = delta_mean - CI_60;
delta_upper_60 = delta_mean + CI_60;
```

%Calculate the predictive deflection based on the forward model and cutoff samples on 90CI and 60CI

```
Deflection_90CI = [];
temp_90_CI = [];
Deflection_60CI = [];
temp_60_CI = [];

for i = 1:n
        %calculate the 90% CI deflection
```

```
        if (E_sample(i) >= E_lower_90) && (E_sample(i) <= E_upper_90) &&
( delta_sample(i) >= delta_lower_90) && (delta_sample(i) <= delta_upper_90)

            temp_90_CI = GroundTruth(E_sample(i),delta_sample(i),1,0);
        else
            temp_90_CI = [];

        end

        Deflection_90CI = [Deflection_90CI; temp_90_CI];

        %calculate the 60% CI deflection

        if (E_sample(i) >= E_lower_60) && (E_sample(i) <= E_upper_60) &&
( delta_sample(i) >= delta_lower_60) && (delta_sample(i) <= delta_upper_60)

            temp_60_CI = GroundTruth(E_sample(i),delta_sample(i),1,0);
        else
            temp_60_CI = [];

        end

        Deflection_60CI = [Deflection_60CI; temp_60_CI];

end
```

Spline curve fitting to smooth the line for the 90CI

```
x = 1:29;%29 measurement position along the beam
y_CI90 = -Deflection_90CI;
for i = 1:size(y_CI90,1)
    P = polyfit(x,y_CI90(i,:),3);
    xi = 1:0.1:29;
    y_poly_CI90(i,:) = polyval(P,xi);
end
```

Loop to fill the error band 90%CI

```
for i = 1:size(y_poly_CI90,1)-1
    hold on;
    fill([xi fliplr(xi)], [y_poly_CI90(i,:) fliplr(y_poly_CI90(i+1,:))], 'cyan',
'FaceAlpha', 1,'EdgeColor','none');
end
hold on;
xlabel('Beam length \it{L} \rm(m)','FontSize',20);
pbaspect([1 0.3 1]);
ylim([-6,0]);
ax = gca;
ax.XAxisLocation = 'top';
```

```
ylabel('Deflection (m)','FontSize',20);
box on;
set(ax,'FontSize',20);
yticks([-6:-1:0]);
yticks('auto');
```

Spline curve fitting to smooth the line for the 60CI

```
x = 1:29;%29 measurement position along the beam
y_CI60 = -Deflection_60CI;
for i = 1:size(y_CI60,1)
    P = polyfit(x,y_CI60(i,:),3);
    xi = 1:0.1:29;
    y_poly_CI60(i,:) = polyval(P,xi);
end
```

Loop to fill the error band 60%CI

```

for i = 1:size(y_poly_CI60,1)-1
    hold on;
    fill([xi fliplr(xi)], [y_poly_CI60(i,:) fliplr(y_poly_CI60(i+1,:))], 'blue',
'FaceAlpha', 1,'EdgeColor','none');
end
hold on;
```

Draw the mean value of the deflection （mean of the 90CI ）

```

for i = 1: size(myData.y,1)
    scatter(x,-myData.y(i,:),'black','x');
    hold on;
end
```

Legend

```
f_mean = plot(xi,mean(y_poly_CI90),'red','LineWidth',2.5);
rectangle('Position', [23, -4,0.5, 0.25], 'FaceColor', 'cyan');
text(24, -3.85, '90% confidence interval', 'FontSize', 12);

rectangle('Position', [23, -4.5,0.5, 0.25], 'FaceColor', 'blue');
text(24, -4.35, '60% confidence interval', 'FontSize', 12);

text(23.1, -4.85, 'x      measurement points', 'FontSize', 12);
hold on;
line([22.9,23.6],[-5.3,-5.3],'linestyle','-','color','red','LineWidth',2.0);
text(24, -5.3, 'mean', 'FontSize', 12);
ax =gca;
exportgraphics(ax,'figure4.pdf','ContentType','vector');
```