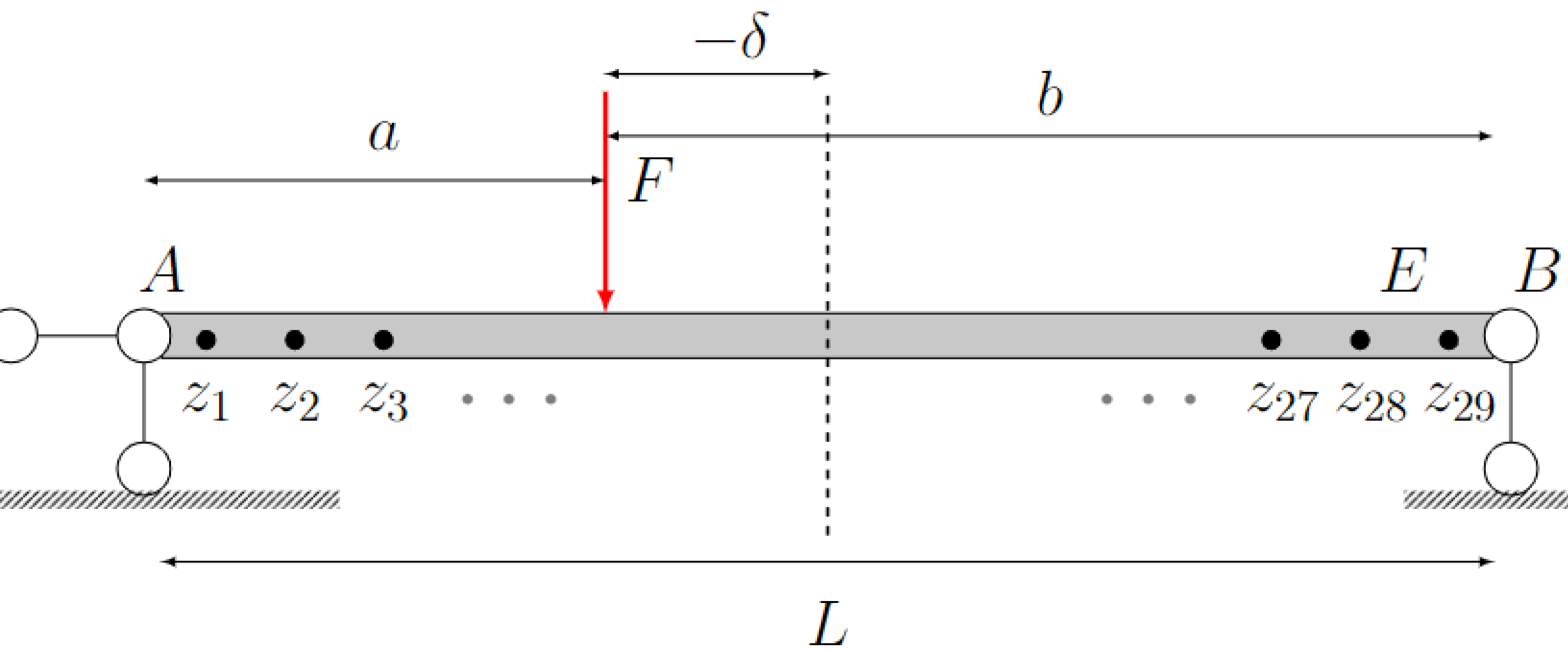
Chain-PCE -- Multiple point-Bayesian inference

1 - INITIALIZE UQLAB

```
clc;clear all;close all;
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:
F:\abaqustemp\UQLab_Rel2.0.0\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 - COMPUTATIONAL MODEL



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)

Computational model:

$$a = \frac{L}{2} - \delta; b = \frac{L}{2} + \delta$$

$$\mathscr{M}(\vec{\theta}) = \frac{Fbz[(L^2 - b^2) - z^2]}{6LEI} \qquad z \leq a$$

$$\mathscr{M}(\vec{\theta}) = \frac{Fb[\frac{L}{b}(z-a)^3 + (L^2 - b^2)]}{6LEI} \qquad z > a$$

$$\vec{\theta} = [E, \delta, z]; \vec{z} = [z_1, z_2, \ldots, z_{28}, z_{29}];$$

E is elastic modulus; *δ* is the loading postion

$\vec{z}$ is the different measurement points along the beam;

$\mathscr{M}$ is the FE model; Y_i is the measurement data; *N* is the number of experiment expNum.

```
%create N sampling and FE realizations
%LHS sampling

% mean of LHS sampling for Gaussian distribution (E and delta)
mu_LHS = [25e9 0];

% sigma and Covariance martrix of LHS sampling for Gaussian distribution (E and delta)
sigma_LHS = [5e9 5].^2;
CovarianceMatrix_LHS = diag(sigma_LHS);

% LHS sampling Number
N = 10;
LHS_sample = lhsnorm(mu_LHS, CovarianceMatrix_LHS, N); %
size(LHS_sample)
```

```
ans = 1×2
    10     2
```

```
E = LHS_sample(:,1);
size(E)
```

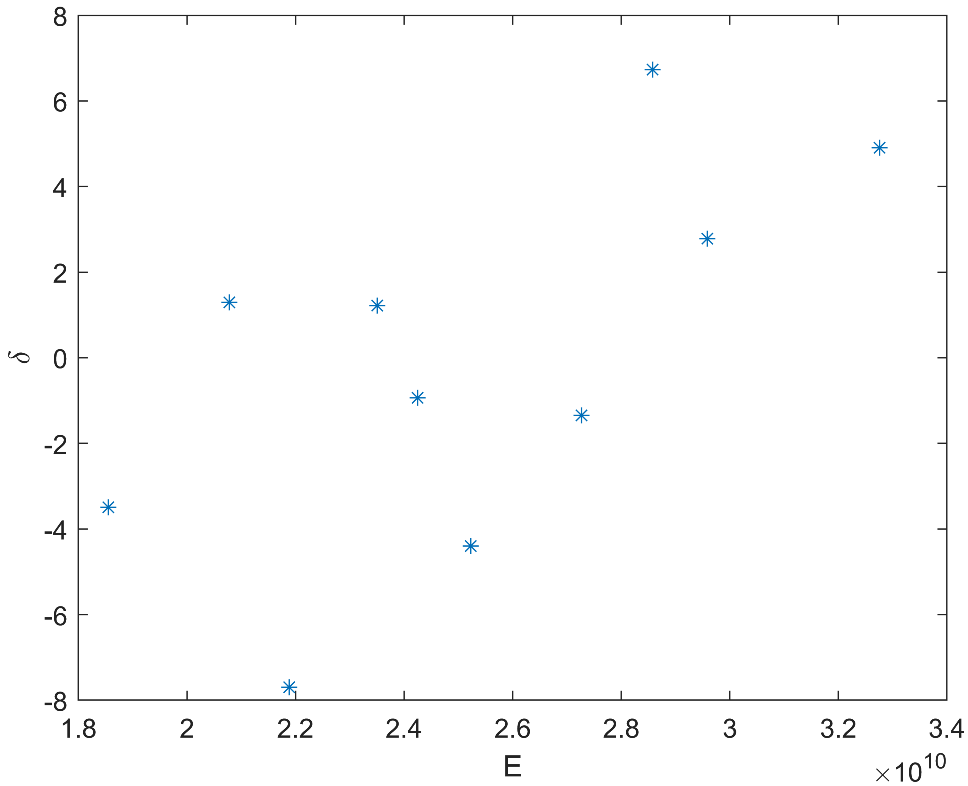
```
ans = 1×2
    10     1
```

```
delta = LHS_sample(:,2);
size(delta)
```

```
ans = 1×2
    10     1
```

```
%plot LHS sampling
figure
```

```
plot(E,delta,'*')
xlabel('E');
ylabel('\delta');
```




```
%FE realizaiton - 29 points deflection along the beam
FE_deflection = Deflection(LHS_sample);
size(FE_deflection)
```

```
ans = 1×2
    10    29
```

```
%FE realizaiton - E + delta + 29 points deflection along the beam
FE_realization = [LHS_sample,FE_deflection];
size(FE_realization)
```

```
ans = 1×2
    10    31
```

3 - PROBABILISTIC INPUT MODEL and POLYNOMIAL CHAOS EXPANSION (PCE) $\widetilde{\mathcal{M}}(r)$

Obtain the number of the Chain N_chain, 29 mulitple targets means 29 chain regressions

```
N_chain = size(FE_realization,2)-2
```

```
N_chain = 29
```

Loop to create N_chain number of PCE $\tilde{\mathcal{M}}(r)$

```
for i = 1:N_chain
```

To create PCE $\tilde{\mathcal{M}}(r)$, input model basis is requried----Adopt uniform distribution to search the space

```
    % create an empty structure to store the input model
    eval(['InputOpts_', num2str(i), ' = struct();']);

     %create mariginals for PCE model
     %loop to assign the maginals
    for j = 1:i+1

        eval(['InputOpts_', num2str(i), '.Marginals(',num2str(j),').Type = ''Uniform'';']);
        eval(['InputOpts_', num2str(i), '.Marginals(',num2str(j),').Parameters = [',num2str(mir
    end

    %create the inputopts for PCE basis for surrogate model

    eval(['myInput_', num2str(i), '= uq_createInput(InputOpts_',num2str(i),')',';']);
```

Chain regression PCE $\tilde{\mathcal{M}}(r)$

```
    %meta options
    metaopts.Type = 'Metamodel';
    metaopts.MetaType = 'PCE';
    metaopts.Method = 'LARS';
    %metaopts.TruncOptions.qNorm = 0.75;
    metaopts.Degree = 1:20;
```

Use experimental design loaded from the data files:

```
    X = FE_realization(:,1:i+1);
    Y = FE_realization(:,i+2);
    metaopts.ExpDesign.X = X;
    metaopts.ExpDesign.Y = Y;
```

Chain regression PCECalculation

```
    eval(['myPCE_', num2str(i), ' = uq_createModel(metaopts',')',';']);
```

export the PCE

```
    eval(['save myPCE_',num2str(i),' ','myPCE_',num2str(i),';']);
end
```

```
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 3 and qNorm 1.00 for output variable 1
Final LOO error estimate: 5.097921e-02
---                 Calculation finished!                         ---
---   Calculating the PCE coefficients by regression.   ---
```

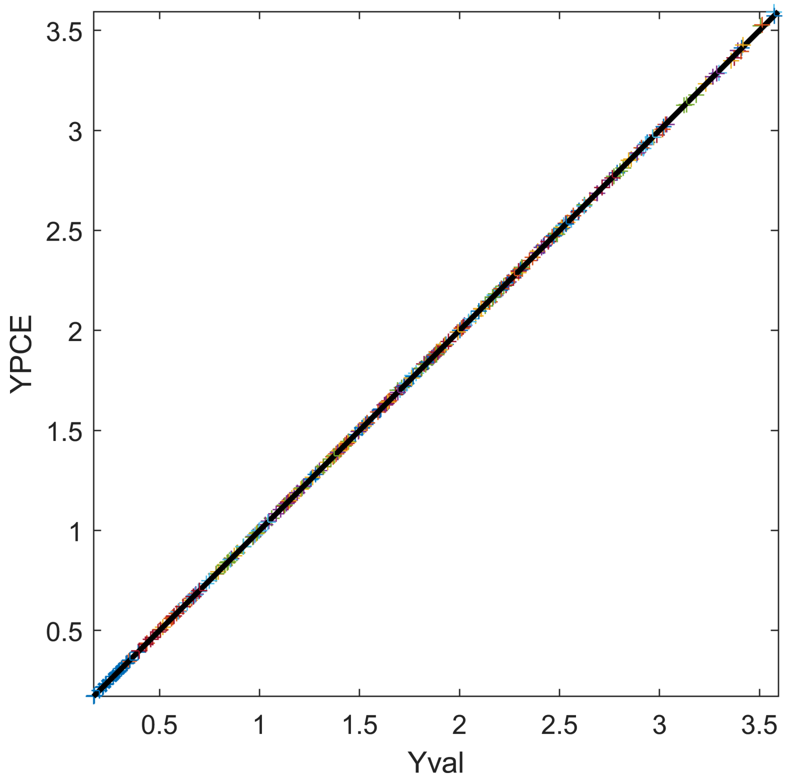
```
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 2.841664e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 8.181284e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.567585e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
警告: Warning: numerical instability!! Gamma for LAR iteration 5 was set to 0 to prevent crashes.
警告: Warning: numerical instability!! Gamma for LAR iteration 6 was set to 0 to prevent crashes.
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 5.359922e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 8.438240e-07
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
警告: Warning: numerical instability!! Gamma for LAR iteration 7 was set to 0 to prevent crashes.
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 3.802932e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 8.652398e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 2.160480e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 2.025293e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 7.093884e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
警告: Warning: numerical instability!! Gamma for LAR iteration 6 was set to 0 to prevent crashes.
警告: Warning: numerical instability!! Gamma for LAR iteration 7 was set to 0 to prevent crashes.
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 9.726456e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.221579e-03
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.322185e-03
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 8.024638e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.114622e-03
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
```

```
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 7.765138e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 3.454943e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 3.968324e-04
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 2 and qNorm 1.00 for output variable 1
Final LOO error estimate: 7.666665e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 8.002719e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 2.598328e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.943050e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.470270e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 1.005490e-05
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 7.347889e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 3.969522e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 2.009890e-06
---                 Calculation finished!                              ---
---   Calculating the PCE coefficients by regression.   ---
The estimation of PCE coefficients converged at polynomial degree 1 and qNorm 1.00 for output variable 1
Final LOO error estimate: 7.691331e-07
---                 Calculation finished!                              ---
```

4 - Yval vs YPCE

```
close all;
for i = 1:N_chain

    Xval = FE_realization(:,1:i+1);
    %size(Xval)
    Yval = FE_realization(:,i+2);
    %size(Yval)
    YPCE = uq_evalModel(eval(['myPCE_', num2str(i)]),Xval);
```

```
    %size(YPCE)
    hold on;

    % plot Yval vs YPCE
    plot(Yval,YPCE,'+');
    hold on;
    plot([min(Yval),max(Yval)],[min(Yval),max(Yval)],'k-','LineWidth',2);

    % plot options
    axis equal
    axis([min(FE_deflection,[],'all') max(FE_deflection,[],'all') min(FE_deflection,[],'all') m


end
xlabel('Yval');
ylabel('YPCE');
box on;
```



5 - Define the priors for E, δ and discrepancy σ

Note: priors for E and δ are different from input models above

By default, UQlab assumes an independent and identically distributed discrepancy

$\varepsilon \sim \mathcal{N}(0, \mu_y^2)$, with $\mu_y = \frac{1}{N} \sum_{i=1}^{N} y_i$

synthetic ground truth with 3% noise

E = 30e9Pa;δ = 4; noise = 3%

```
Measurement = GroundTruth(30e9,4,1,0.03);
size(Measurement)
```

```
ans = 1×2
     1    29
```

priors

```
%Priors on E , delta and sigma
PriorOpts.Marginals(1).Name = 'E';                % Young's modulus
PriorOpts.Marginals(1).Type = 'Gaussian';
PriorOpts.Marginals(1).Parameters = [25e9 5e9];   % (N/m^2)
PriorOpts.Marginals(1).Bounds = [10e9 35e9];

PriorOpts.Marginals(2).Name = 'delta';                % Concentrated load loading position
PriorOpts.Marginals(2).Type = 'Gaussian';
PriorOpts.Marginals(2).Parameters = [0 5]; % (N/m)
PriorOpts.Marginals(2).Bounds = [-10  10];


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);

% SigmaOpts.Marginals(1).Name = 'Sigma2';
% SigmaOpts.Marginals(1).Type = 'Uniform';
% sigma2 = mean(Measurement(:,:),"all");
% SigmaOpts.Marginals(1).Parameters = [0 sigma2.^2];
%
% mySigmaDist = uq_createInput(SigmaOpts);
% DiscrepancyOptsUnknownDisc.Type = 'Gaussian';
% DiscrepancyOptsUnknownDisc.Prior = mySigmaDist;
```

6 - Define the custom-loglikelihood and measurement data for UQlab calculation

$$\ell\mathscr{L}(\vec{\theta},\epsilon \mid Y) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{3/2}\det(\Sigma(\epsilon))^{1/2}} \exp\left(-\frac{1}{2}\left(Y_i - \mathscr{M}(\vec{\theta})\right)^T \Sigma(\epsilon)^{-1}\left(Y_i - \mathscr{M}(\vec{\theta})\right)\right)$$

```
myData.y = Measurement;
myData.Name = 'Measurement on 29 points along the beam';
```

Loglikelihood still follows the Gaussian discrepancy criteria

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

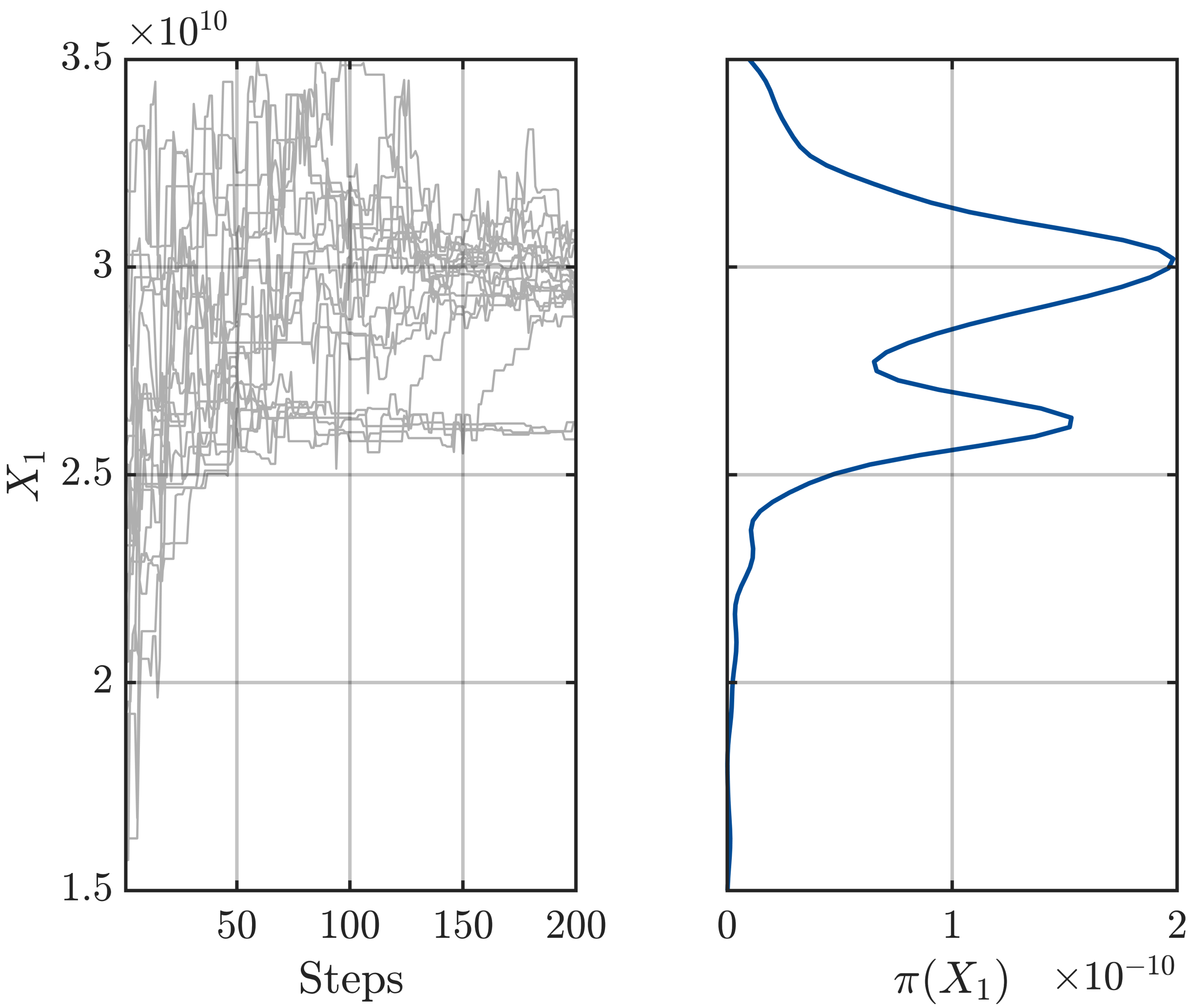
7 - Solver options

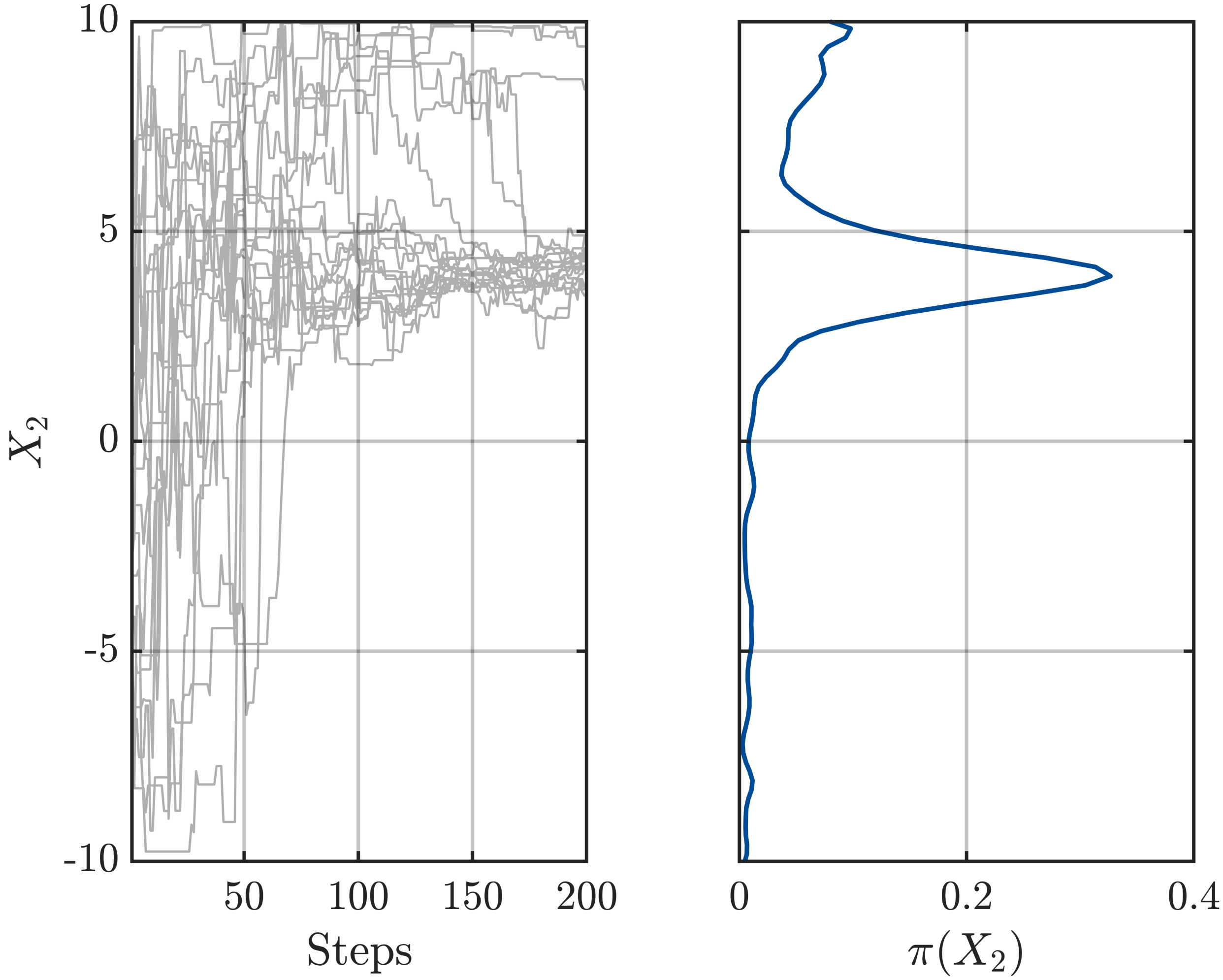
```
Solver.Type = 'MCMC';
Solver.MCMC.Visualize.Parameters = [1 2];
Solver.MCMC.Visualize.Interval = 10;
Solver.MCMC.Sampler = 'AIES';
Solver.MCMC.Steps = 200;
Solver.MCMC.NChains = 20;
Solver.MCMC.Proposal.PriorScale = 1e-3;
```

8 - Bayesian inference

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

```
Starting AIES...

|##                              |   5.00%
```






```
|##############################| 100.00%

Finished AIES!
```

9 - Post-processing

Burn in 70%; badchain criteria $\delta > 10m$; confidence interval 90%; Point estimate - mean

```
badChainsIndex = squeeze(BayesAnalysis.Results.Sample(end,2,:) > 10);

uq_postProcessInversionMCMC(BayesAnalysis,'pointEstimate','mean','percentiles',[0.05,0.95],'bur


uq_print(BayesAnalysis);
```

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

   Algorithm:                                     AIES
   Duration (HH:MM:SS):                           01:17:56
   Number of sample points:                       4.00e+03
```

```
%------------------- Posterior Marginals
---------------------------------------------------------------
| Parameter | Mean    | Std     | (0.05-0.95) Quant.    | Type  |
---------------------------------------------------------------
| E         | 2.9e+10 | 1.6e+09 | (2.6e+10 - 3.1e+10) | Model |
| delta     | 5       | 2.1     | (3.4 - 9.9)           | Model |
| sigma2    | 0.0045  | 0.0058  | (0.0013 - 0.012)      | Model |
---------------------------------------------------------------

%------------------- Point estimate
----------------------------------------
| Parameter | mean    | Parameter Type |
----------------------------------------
| E         | 2.9e+10 | Model          |
| delta     | 5       | Model          |
| sigma2    | 0.0045  | Model          |
----------------------------------------

%------------------- Correlation matrix (model parameters)
----------------------------------------
|        |  E         delta     sigma2  |
----------------------------------------
| E      |  1         -0.86     -0.088  |
| delta  |  -0.86     1         0.47    |
| sigma2 |  -0.088    0.47      1       |
----------------------------------------
```

```
%uq_display(BayesAnalysis);
```

10 - 90% error band on predictive posterior

90% confidence interval for E and δ

set 90% = 95%- 5%

```
uq_postProcessInversionMCMC(BayesAnalysis,'percentiles',[0.05,0.95]);
```

Obtained the lower bound and upper bound for E and δ

```
E_5_LowB = BayesAnalysis.Results.PostProc.Percentiles.Values(1,1);
size(E_5_LowB)
```

```
ans = 1×2
     1     1
```

```
E_95_UpperB = BayesAnalysis.Results.PostProc.Percentiles.Values(2,1);
size(E_95_UpperB)
```

```
ans = 1×2
     1     1
```

```
delta_5_LowB = BayesAnalysis.Results.PostProc.Percentiles.Values(1,2);
size(delta_5_LowB)
```

```
ans = 1×2
     1     1
```

```
Delta_95_UpperB = BayesAnalysis.Results.PostProc.Percentiles.Values(2,2);
size(Delta_95_UpperB)
```

```
ans = 1×2
     1     1
```

Sampling on predictive distribution

```
N_predict = 10
```

```
N_predict = 10
```

```
% sampling on E
E_90_sample_O = linspace(E_5_LowB,E_95_UpperB,N_predict)';
size(E_90_sample_O)
```

```
ans = 1×2
    10     1
```

```

%Shuffle the order
shuffledIndices = randperm(length(E_90_sample_O));
E_90_sample = E_90_sample_O(shuffledIndices);
size(E_90_sample)
```

```
ans = 1×2
    10     1
```

```
% sampling on delta
delta_90_sample_O = linspace(delta_5_LowB,Delta_95_UpperB,N_predict)';
size(delta_90_sample_O)
```

```
ans = 1×2
    10     1
```

```
%Shuffle the order
shuffledIndices = randperm(length(delta_90_sample_O));
delta_90_sample = delta_90_sample_O(shuffledIndices);
size(delta_90_sample)
```

```
ans = 1×2
    10     1
```

```

%plot the sampling on E and delta
figure
plot(E_90_sample,delta_90_sample,'*');
xlabel('E');
ylabel('\delta');

axis([min(E_90_sample,[],'all') max(E_90_sample,[],'all') min(delta_90_sample,[],'all') max(del
```

Predictive FE realization

```
Predict_sample = [E_90_sample,delta_90_sample];
size(Predict_sample)
```

```
ans = 1×2
    10     2
```

```
%Loop to get the predictive FE deflection
YPCE_Predict = [];

for i = 1:N_predict

    Xval_Predict = Predict_sample(i,:);

    YPCE_OneRowAllPoint_Predict = [];

    for j = 1:N_chain

        eval(['YPCE_OneRowOnePoint = uq_evalModel(myPCE_', num2str(j),',Xval_Predict)',';']);
        YPCE_OneRowAllPoint_Predict = [YPCE_OneRowAllPoint_Predict,YPCE_OneRowOnePoint];
        Xval_Predict = [Xval_Predict,YPCE_OneRowOnePoint];
    end

    YPCE_Predict = [YPCE_Predict;YPCE_OneRowAllPoint_Predict];


end


size(YPCE_Predict)
```

```
ans = 1×2
    10    29
```

Spline curve fitting to smooth the line for the 90CI

```
x = 1:29;%29 measurement position along the beam

for i = 1:size(YPCE_Predict,1)

    P = polyfit(x,YPCE_Predict(i,:),3);
    xi = 1:0.1:29;
    YPCE_Predict_Poly(i,:) = polyval(P,xi);
end
size(YPCE_Predict_Poly)
```

```
ans = 1×2
    10   281
```

Loop to fill the error band 90%CI

```
%loop to fill the error band
close all;
for i = 1:size(YPCE_Predict_Poly,1)-1
    hold on;
    fill([xi fliplr(xi)], [-YPCE_Predict_Poly(i,:) fliplr(-YPCE_Predict_Poly(i+1,:))], 'cyan',
end

 hold on;
```

```
xlabel('Beam length \it{L} \rm(m)','FontSize',10);
pbaspect([1 0.3 1]);
ax = gca;
ax.XAxisLocation = 'top';
ylabel('Deflection (m)','FontSize',10);
box on;
set(ax,'FontSize',10);
yticks('auto');
ylim([-3.5 0])
```

scatter the measurement

```
x = 1:1:29;
size(x)
```

```
ans = 1×2
     1    29
```

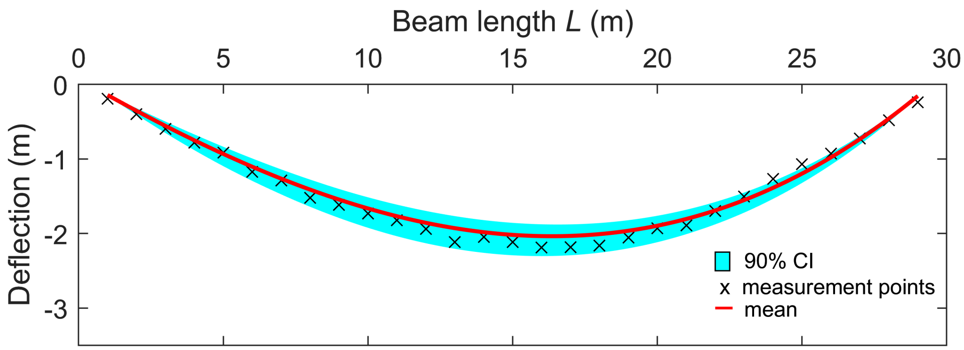
```
for i = 1: size(myData.y,1)
    scatter(x,-myData.y(i,:),'black','x');
    hold on;
end
```

draw the mean value of 90%CI

```
plot(xi,-mean(YPCE_Predict_Poly),'red','LineWidth',1.5);
% beam position
hold off;
```

legend

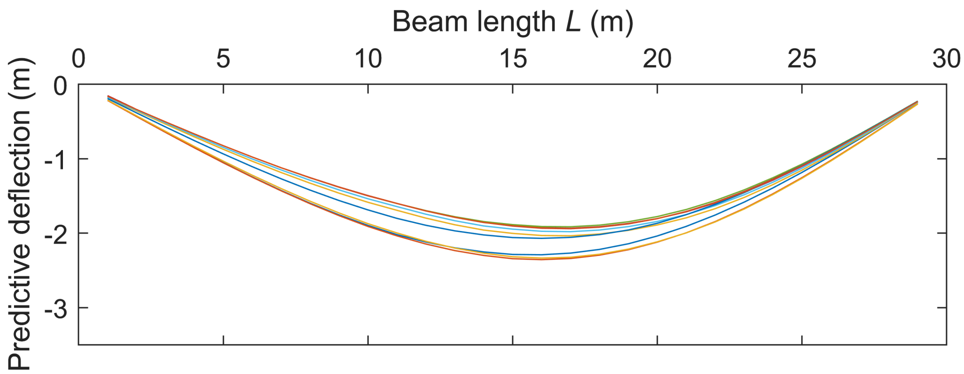
```
rectangle('Position', [22, -2.5,0.5, 0.25], 'FaceColor', 'cyan');
text(23, -2.35, '90% CI', 'FontSize', 8);
text(22.15, -2.7, 'x  measurement points', 'FontSize', 8);

line([22,22.6],[-3,-3],'linestyle','-','color','red','LineWidth',1.0);
text(23, -3, 'mean', 'FontSize', 8);
```




```
x_beam = 1:1:29
```

```
x_beam = 1×29
     1     2     3     4     5     6     7     8     9    10    11    12    13 ···
```

```
for i = 1:size(YPCE_Predict,1)

    plot(x_beam,-YPCE_Predict(i,:));
    hold on;
end
 xlabel('Beam length \it{L} \rm(m)','FontSize',10);
 pbaspect([1 0.3 1]);
 ax = gca;
 ax.XAxisLocation = 'top';
 ylabel('Predictive deflection (m)','FontSize',10);
 box on;
 set(ax,'FontSize',10);
 yticks('auto');
  yticks('auto');
 ylim([-3.5 0])
```


Beam length L (m)
0
5
10
15
20
25
30
Predictive deflection (m)
0
-1
-2
-3