Uncertain Modeling in RCT

An uncertain model consisting of a second-order underdamped system with real parameteric uncertainty and high-frequency unmodeled dynamics will be constructed using the `ureal` and `ultidyn` uncertain elements.

UC Berkeley, ME C231B/EECS C220C, Spring 2017

Contents

Uncertain real parameters: `ureal` objects

Create three uncertain real parameter objects.

```
p1 = ureal('p1',4)
```

```
p1 =

  Uncertain real parameter "p1" with nominal value 4 and variability [-1,1].
```

```
p2 = ureal('p2',10,'range',[8 14])
```

```
p2 =

  Uncertain real parameter "p2" with nominal value 10 and range [8,14].
```

```
p3 = ureal('p3',6,'percentage',[-30 40])
```

```
p3 =

  Uncertain real parameter "p3" with nominal value 6 and variability [-30,40]%.
```

Questions to consider.

- What is the `Range` of `p1`
- If the `NominalValue` of `p1` is changed to 3, what is the `Range` of `p1`

```
p1.NominalValue = 3 % cannot change the read only field
p1 = ureal('p1',3) % construct a new uncertain real parameter, the range is
% still [-1,1]
% * If the |NominalValue| of |p2| is changed to 11, what is the |Range| of
% |p2|
p2 = ureal('p2',11,'range',[8 14]) % the nominal value is changed to 11 but
% the range keeps the same
% * If the |NominalValue| of |p3| is changed to 10, what is the |Range| of
% |p3|
%
p3 = ureal('p3',10,'percentage',[-30 40]) % the nominal value can be changed
% to 10 and the range keeps the same
% Experiment at the command line to answer these questions, and confirm
% your understanding in how the objects behave.
```

```
p1 =

  Uncertain real parameter "p1" with nominal value 3 and variability [-1,1].


p1 =

  Uncertain real parameter "p1" with nominal value 3 and variability [-1,1].


p2 =

  Uncertain real parameter "p2" with nominal value 11 and range [8,14].
```

```
p3 =

  Uncertain real parameter "p3" with nominal value 10 and variability [-30,40]%.
```

Real Parametric Uncertainty

Use `ureal` objects to construct a damping ratio with nominal value of 0.1 and 35% uncertainty.

```
zeta = ureal('zeta',0.1,'Percentage',35)
```

```
zeta =

  Uncertain real parameter "zeta" with nominal value 0.1 and variability [-35,35]%.
```

Next, construct an uncertain natural frequency with nominal value of 4 rad/sec and an uncertain range of 3.3 to 5 rad/sec. Examine the properties of this object.

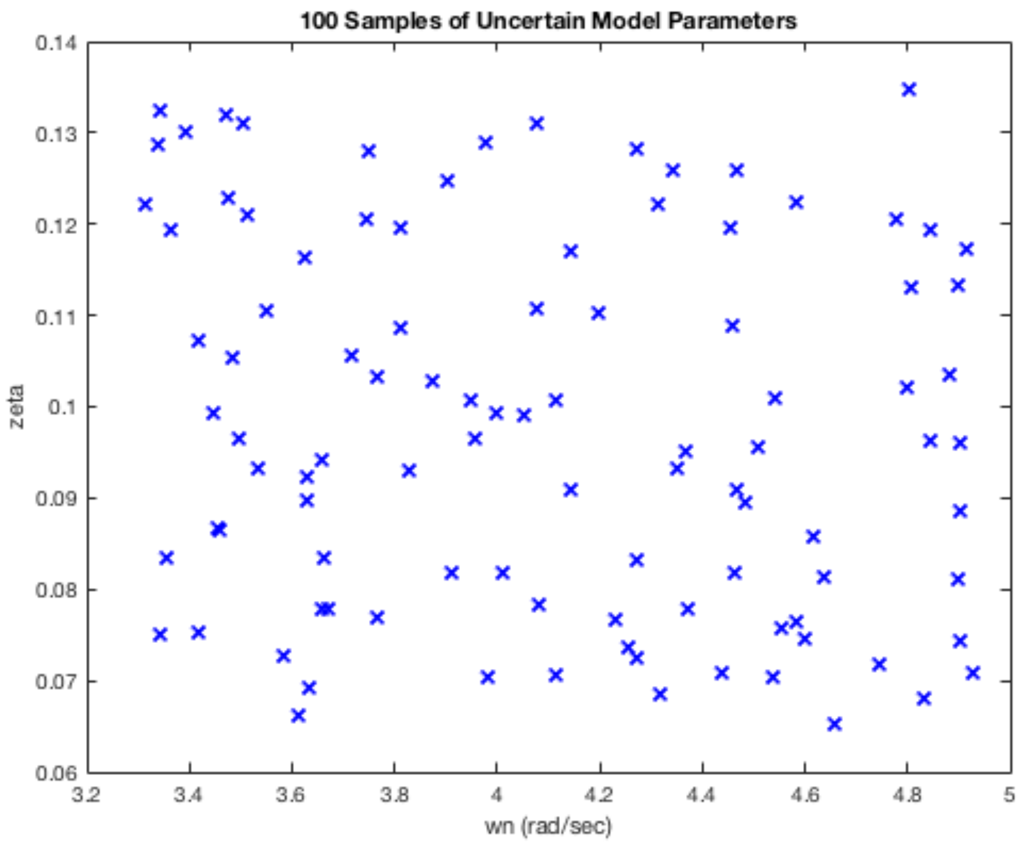
```
wn = ureal('wn',4,'Range',[3.3 5]);
get(wn)
```

```
    NominalValue: 4
            Mode: 'Range'
           Range: [3.3000 5]
       PlusMinus: [-0.7000 1]
      Percentage: [-17.5000 25]
    AutoSimplify: 'basic'
            Name: 'wn'
```

Sample Parametric Uncertainty

The `usample` command can be used to generate random samples of the real parameteric uncertainty defined above.

```
RSamples = usample([wn;zeta],100);
plot(RSamples(1,:),RSamples(2,:),'bx');
xlabel('wn (rad/sec)');
ylabel('zeta');
title('100 Samples of Uncertain Model Parameters');
```



Uncertain matrices

Create three real parameters.

```
a = ureal('a',2);
b = ureal('b',0.5,'Perc',30);
c = ureal('c',10,'range',[7 11]);
```

Use standard Matlab manipulations to concatenate these objects, along with numerical values into an uncertain matrix.

```
G = [a b; a*c b-c; 4+a c^2]
```

```
G =

  Uncertain matrix with 3 rows and 2 columns.
  The uncertainty consists of the following blocks:
    a: Uncertain real, nominal = 2, variability = [-1,1], 2 occurrences
    b: Uncertain real, nominal = 0.5, variability = [-30,30]%, 1 occurrences
    c: Uncertain real, nominal = 10, range = [7,11], 4 occurrences

Type "G.NominalValue" to see the nominal value, "get(G)" to see all properties, and "G.Uncerta
inty" to interact with the uncertain elements.
```

```
size(G)
```

```
Uncertain matrix with 3 rows, 2 columns, and 7 blocks.
```

```
class(G)
```

```
ans =

    'umat'
```

```
G.Uncertainty.b.NominalValue = 1;
```

```
G.Uncertainty.b.Range
```

```
ans =

    0.7000    1.3000
```

Confirm that the uncertain element named `'b'` within G is not a pointer to the workspace variable `b`.

```
b.Range
```

```
ans =

    0.3500    0.6500
```

Sampling and Substitution

Sample 150 points from the uncertainty space of G

```
[Gs,S] = usample(G,150);
```

```
size(Gs)
```

```
ans =

     3     2   150
```

```
S
```

```
S = 

  150×1 struct array with fields:

    a
    b
    c
```

Sample 15 points from a rectangular grid, formed by 15 points in the uncertainty space spanned by `'a'` and `'b'`, and 10 points from the space spanned by `'c'`.

```
[Gs,S] = usample(G,{'a' 'b'},15,'c',10);
size(Gs)
```

```
ans =

     3     2    15    10
```

The samples are stored in `S`, and can used for substitution into any other object which has dependence on `'a'`, `'b'` and/or `'c'`. Here, for illustrative purposes, recover `Gs` from the samples

```
Gsrecover = usubs(G,S);
norm(Gs(:)-Gsrecover(:))  % floating point roundoff in substitution
```

```
ans =

   1.0091e-13
```

Substitute specific values for `'a'`, `'b'` and `'c'`

```
usubs(G,'a',1,'b',2,'c',3)
```

```
ans =

    1.0000    2.0000
    3.0000   -1.0000
    5.0000    9.0000
```

Dynamic Uncertainty

Create an uncertain linear time-invariant object, `Delta`. By default, all `ultidyn` objects are norm bounded by 1 at all frequency.

```
iosize = [1 1];
Delta = ultidyn('Delta',iosize);
```

```
Delta.Type
```

```
ans =

    'GainBounded'
```

```
Delta.Bound
```

```
ans =

     1
```

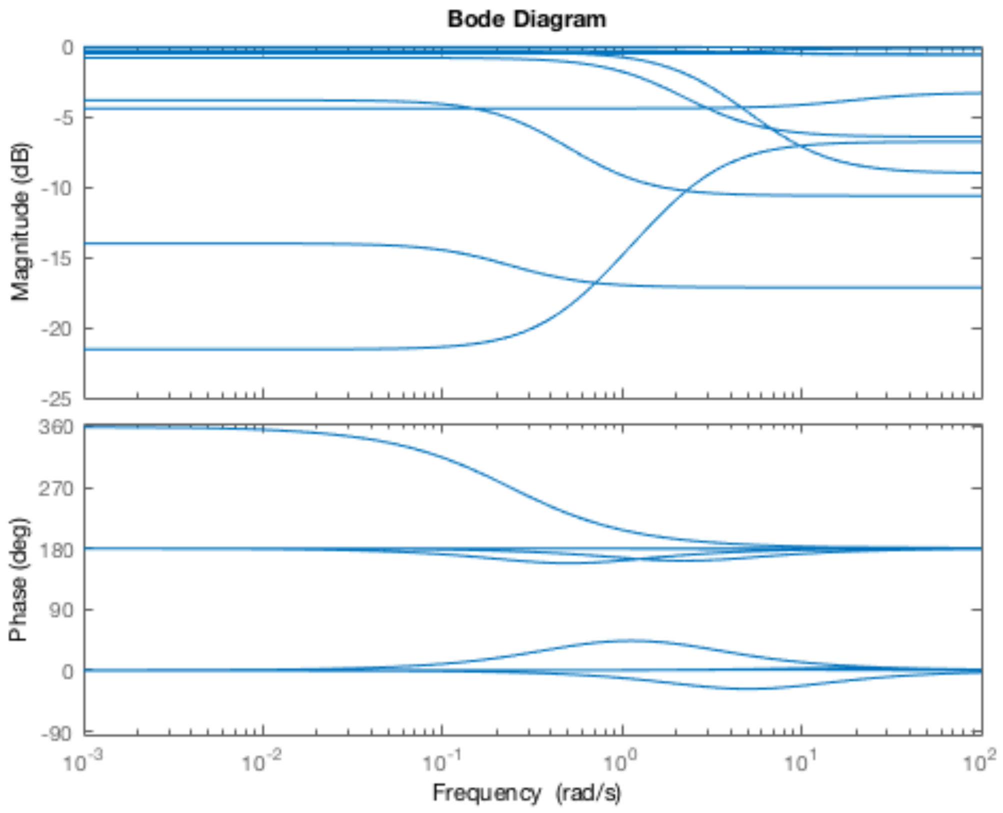
Sample Dynamic Uncertainty

The `usample` command can be used to generate random instances of the uncertain objects. For example, each instance of `Delta` is stable and has state dimension equal to 1, as specified by the `SampleStateDim` property.

```
Delta.SampleStateDim
```

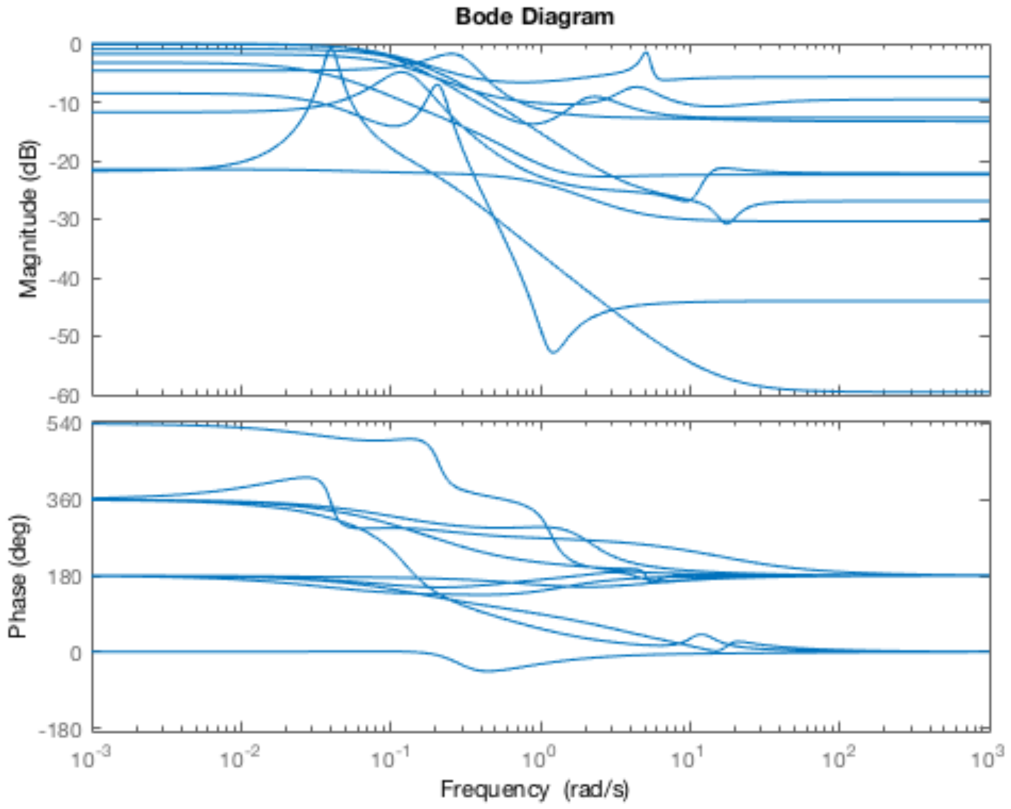
```
ans =

     1
```

```
nSamples = 10;
DSamples = usample(Delta, nSamples);
bode(DSamples);
```



The state dimension of the uncertain instances can be modified using the `SampleStateDim` property. Change this property value so that subsequent samples will be of state dimension 5.

```
Delta.SampleStateDim = 5;
DSamples2 = usample(Delta, nSamples);
bode(DSamples2);
```



Uncertain systems using `tf`

```
tau = ureal('tau',0.5,'Perc',[-50 100]);
gamma = ureal('gamma',1,'Perc',[-30 60]);
Delta = ultidyn('Delta',[1 1],'Type','GainBounded');
W = makeweight(0.2,10,20);
P = tf(gamma,[tau 1])*(1 + W*Delta)
```

```
P =

  Uncertain continuous-time state-space model with 1 outputs, 1 inputs, 2 states.
  The model uncertainty consists of the following blocks:
    Delta: Uncertain 1x1 LTI, peak gain = 1, 1 occurrences
    gamma: Uncertain real, nominal = 1, variability = [-30,60]%, 1 occurrences
    tau: Uncertain real, nominal = 0.5, variability = [-50,100]%, 1 occurrences

Type "P.NominalValue" to see the nominal value, "get(P)" to see all properties, and "P.Uncerta
inty" to interact with the uncertain elements.
```

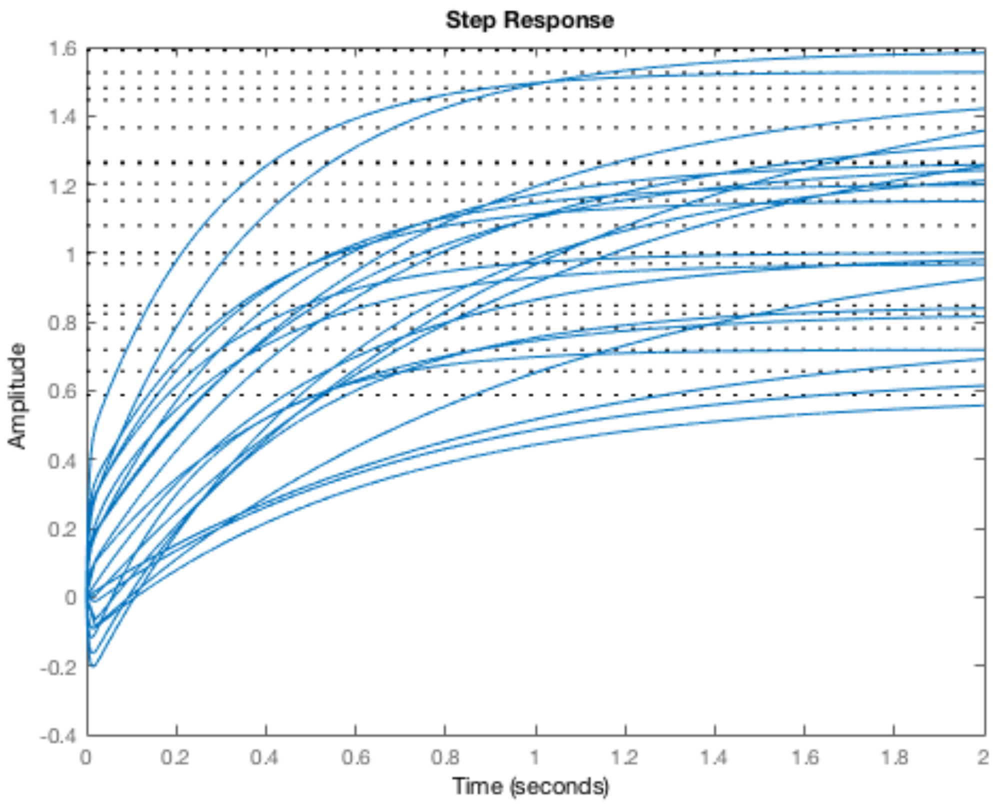
Although `P` is an uncertain state-space object (the is no uncertain transfer function class), the nominal value of `P` can be converted back to a transfer function object.

```
tf(P.NominalValue)
```

```
ans =

    2
  -----
  s + 2

Continuous-time transfer function.
```

The step response is computed on 20 samples, as well as the nominal value.

```
step(P, 2)
```

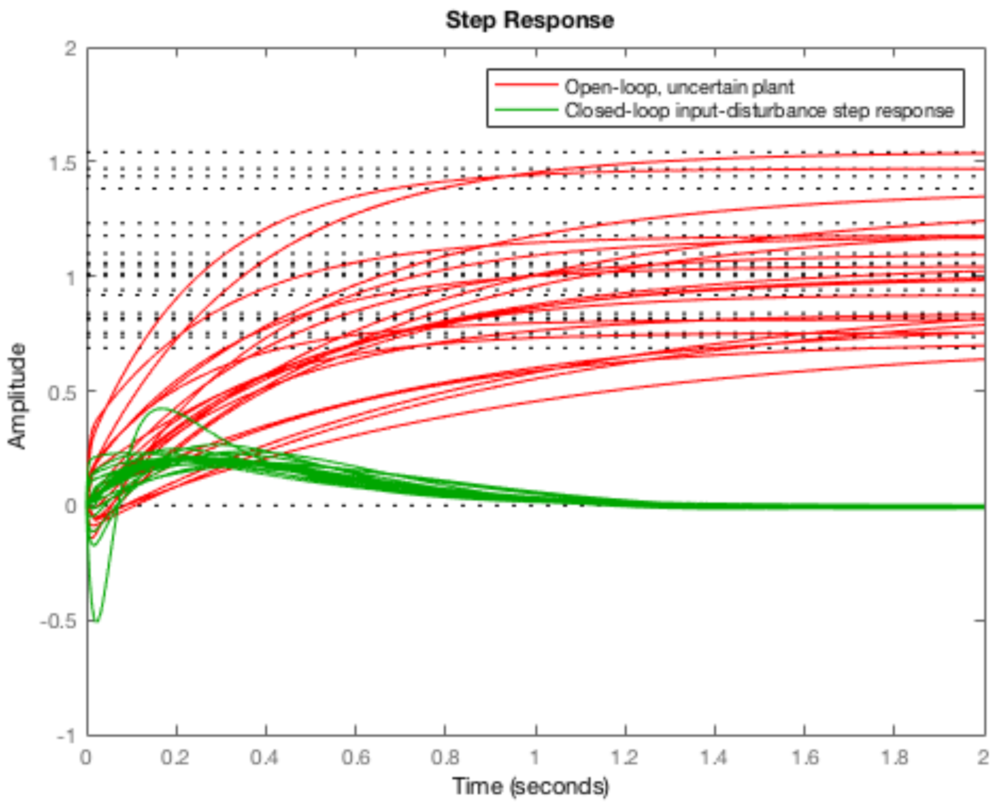


`feedback` for simple closed-loop systems

A simple PI controller is effective on the uncertain plant

```
K = tf([2.4 8],[1 0]);
```

```
T = feedback(P,K);
step(P,'r', T,'g', 2);
legend('Open-loop, uncertain plant','Closed-loop input-disturbance step response');
```



Build Uncertain Model using uncertain elements

Construct the 2nd order system with uncertain damping ratio and natural frequency. In addition, include a multiplicative uncertainty to model overall uncertainty in the dynamics. The multiplicative weight `Wmult` models 10% dynamic uncertainty at low frequencies, 100% at 10 rad/TimeUnit, and 1000% uncertainty at high frequencies.

```
Wmult = makeweightb(0.1,10,10);
G = tf(wn^2,[1 2*zeta*wn wn^2])*(1+Wmult*Delta)
```

```
Undefined function 'makeweightb' for input arguments of type 'double'.

Error in UncertainModeling (line 159)
Wmult = makeweightb(0.1,10,10);
```

Plot the magnitude of samples of the uincertain model, as well as the nominal model, and the multiplicative uncertainty weight.

```
bodemag(G,'b', G.Nominal,'r', Wmult,'g')
```

```
legend('Nominal model','Multiplicative Uncertainty','location','best');
title('Nominal G and Multiplicative Uncertainty Weight (% uncertainty)')
```

Uncertain State-space models

Create several Uncertain parameters and uncertain matrices.

```
p = ureal('p',10,'Percentage',10);
r = ureal('r',-1,'Percentage',30);
A = [r p;-p r];  % UMAT
B = eye(2);      % DOUBLE
C = [1 p;-p 1];  % UMAT
```

Use the `ss` method to pack these uncertain matrices into an uncertain state-space model.

```
H = ss(A,B,C,0)
```

Compute the poles of 50 samples of `H`, and plot a scatter plot.

```
pH = pole(usample(H,50));
plot(pH(:),'x')
```

A step response of `H` shows the modest variability in frequency and damping.

```
step(H,3)
```

Add high-frequency, unmodeled dynamics to `H`, using a `ultidyn` element, and a weighting function `W`. Note that G has dependence on 3 uncertain elements

```
Delta = ultidyn('Delta',[2 2],'SampleStateDim',4); %default: GainBounded, Bound = 1
W = makeweight(0.2, 50, 20);  % W(j0)=0.2, |W(j50)|=1, W(j?)=20
G = H*(eye(2)+W*Delta)
```

The step response of G is similar to `H`, but has noticably more variability on short time-scales (high frequency).

```
step(G,3)
```

Sample Bode Plots

Generate Bode plots for 15 instances of the random model. The nominal Bode plots are shown for comparison.

```
GSamples = usample(G,15);
bodemag(GSamples,'b', G.Nominal,'r', {1e-1,1e3})
ylim([-60 20]);
title('Bode Magnitude Plot of 15 samples, and Nominal')
```

Insight into the data structure

Create an uncertain matrix from the existing `ureal` objects.

```
G = [p1  p1*p2; p2  p3];
% Manipulate, using typical Matlab commands and row/column referencing and
% assignment syntax.
Gext = [inv(G)  G];
Row1 = G(1,:)
G(3,:) = [5 p1*p2*p3];
```

The uncertain matrix can be decomposed into it's constituent parts.

```
[M,DeltaN] = lftdata(G); % lft(DeltaN,M) equals G
class(M)
class(DeltaN)
Resid = G - lft(DeltaN,M);
usample(Resid,4)
```

Conclusions

The uncertain objects `ureal` and `ultidyn` can be used to form uncertain matrices and uncertain state-space models. Many of the usual operations and manipulations for numerical matrices and state-space models are implemented, and behave as expected.

File Information

```
disp(mfilename)
```

Lessons learned

This tutorial teaches me how to create uncertain objects by specifying their nominal value and range or percentage of variability. Additionally, how to sample certain number of points from the uncertain objects created. Additionally, how to use the ultidyn function to generate uncertain linear time invariant dynamical system, and use the usample to extract certain number of sample variables from the uncertain dynamical system. Last but not the least, how to use the lftdata to decompose an unceratin system into its consitituents, the nominal part and the uncertain parts. And how to use the lft to form back the original uncertain system with thenominal part and the uncertain part.

Published with MATLAB® R2017b