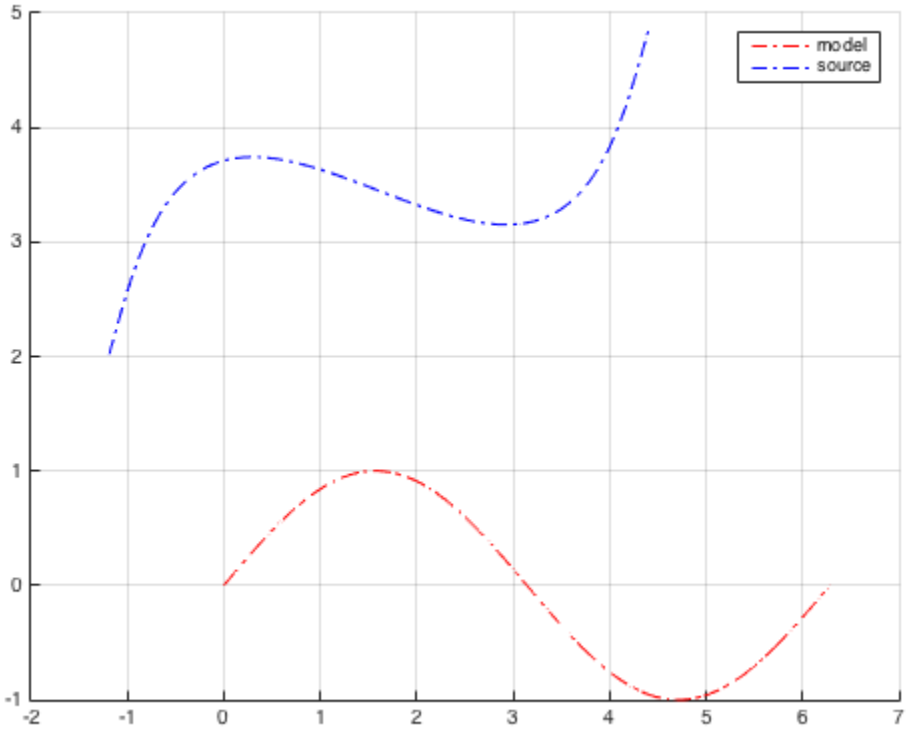
PROJECT 1: ICP Algorithm

PROBLEM1: The ICP algorithm for 2D LINE DATA WITHOUT NOISE

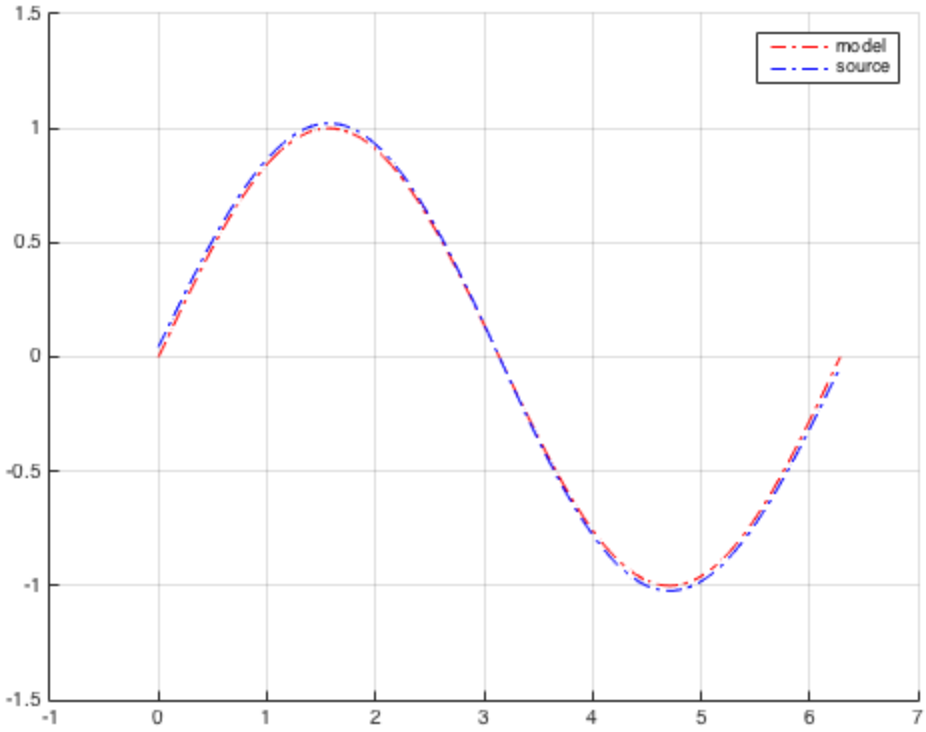
```
clear all;
close all;
%Model data and source data assigning for 2d  data
a = load('2D_line.mat');
mod = a.model;
source = a.source;
figure(1)
hold on
grid on
plot(mod(1,:),mod(2,:),'LineStyle', '-.','Color','r');
plot(source(1,:),source(2,:),'LineStyle', '-.','Color','b');
legend('model','source');
hold off
```



k iteration for the algorithm - ICP

```
for k = 1:1:20
[m,n] = size(mod);
[ms,ns] = size(source);
v = zeros(1,ns);
```

```
diff = zeros(1,ns);
% closest point algorithm
for j = 1:1:ns
        mval = 9e99;
        val =sqrt(sum((mod - repmat(source(:,j),1,n)).^2));
        if val<=mval
            [minim,v(j)] = min(val);
        end
end
 modchanged = mod(:,v);
% application of Principal component analysis for finding the rotation
% matrix
 centroidmod = mean(modchanged,2);% Centroid Model
centroidsource = mean(source,2);%Centroid Source
%Cov(x) = E(xy) - 3*E(x)*E(y)
cov = source* modchanged' - 3*centroidsource*centroidmod';%covariance
 Matrix
[U,~,V]=svd(cov);%singular Value decomposition
Ri=V*U';%Calculating the rotation matrix
T = centroidmod - Ri*centroidsource;%Calculating the translation
 Matrix
Changedpossource = Ri*source + repmat(T,1,ns);%Changing the position
 of the source data
source = Changedpossource;
end
% plotting the data
figure(2)
hold on
grid on
plot(mod(1,:),mod(2,:),'LineStyle', '-.','Color','r');
plot(source(1,:),source(2,:),'LineStyle', '-.','Color','b');
legend('model','source');
hold off
```

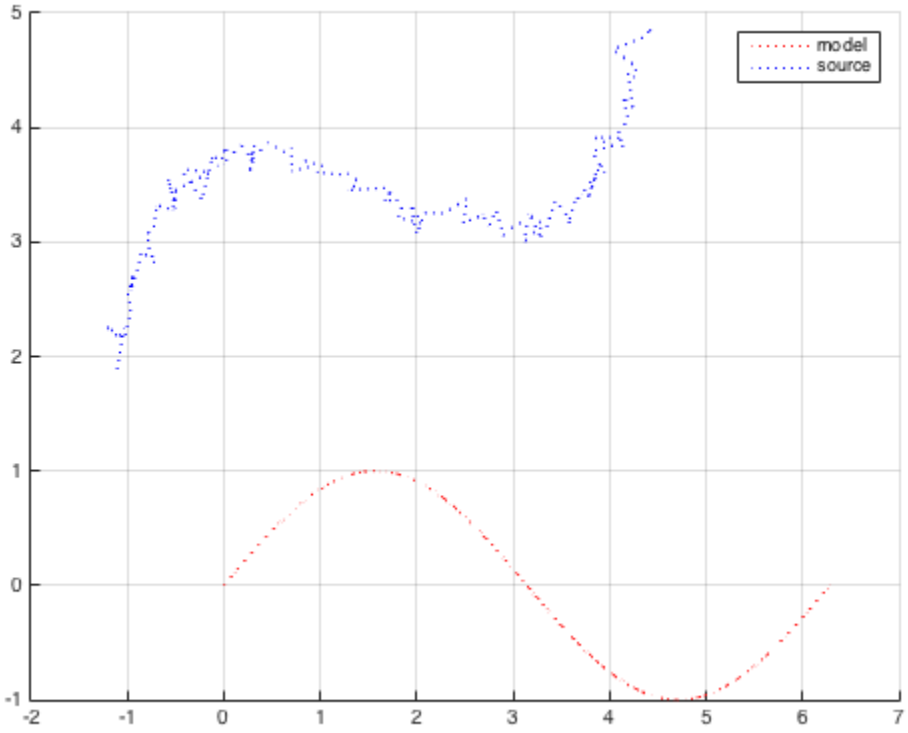


Published with MATLAB® R2015a

Table of Contents

Problem2: The ICP algorithm for 2D LINE DATA WITH NOISE

```
clear all;
close all;
% Model data and source data assigning for 2d data noise
a = load('2D_line_noise.mat');
mod = a.model;
source = a.source;
figure(1)
hold on
grid on
plot(mod(1,:),mod(2,:),'LineStyle', ':','Color','r');
plot(source(1,:),source(2,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off
```

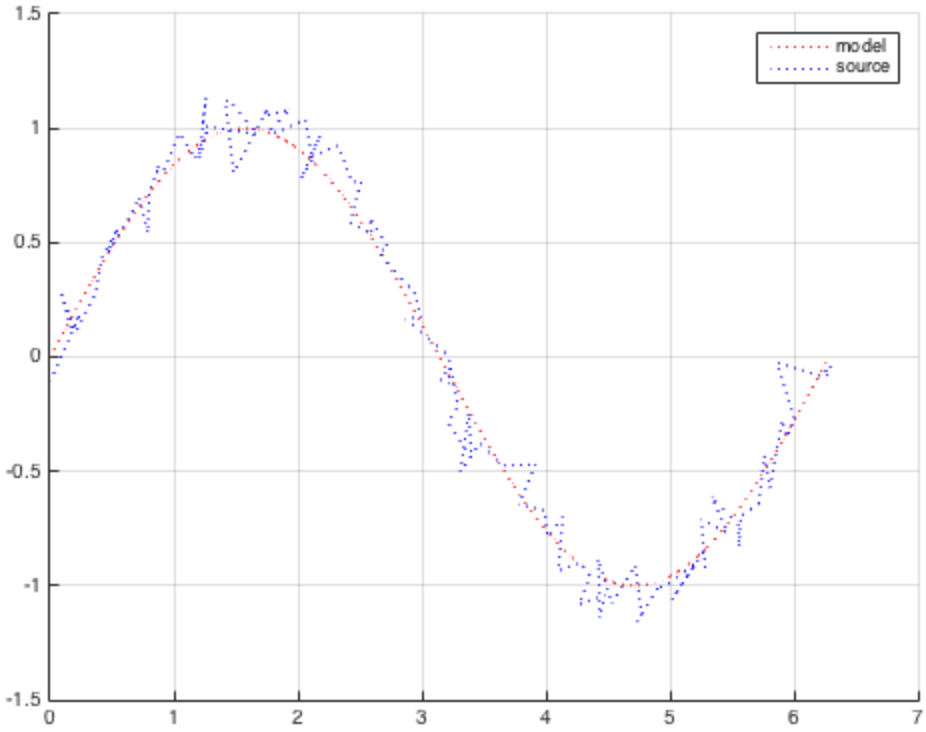


k iteration for the algorithm - ICP

```
for k = 1:1:20
[m,n] = size(mod);
[ms,ns] = size(source);
v = zeros(1,ns);
diff = zeros(1,ns);
% closest point algorithm
for j = 1:1:ns
        mval = 9e99;
        val =sqrt(sum((mod - repmat(source(:,j),1,n)).^2));
        if val<=mval
            [minim,v(j)] = min(val);
        end
end
 modchanged = mod(:,v);
% application of Principal component analysis for finding the rotation
% matrix
 centroidmod = mean(modchanged,2);
centroidsource = mean(source,2);
%Cov(x) = E(xy) - 3*E(x)*E(y)
cov = source* modchanged' - 3*centroidsource*centroidmod';%Covariance
 Matrix Calculation
[U,~,V]=svd(cov);%Singular Value decomposition
Ri=V*U';%Rotation Matrix
T = centroidmod - Ri*centroidsource;%translation vector
Changedpossource = Ri*source + repmat(T,1,ns);%position of the source
 changed
source = Changedpossource;
end
```

plotting the data

```
figure(2)
hold on
grid on
plot(mod(1,:),mod(2,:),'LineStyle', ':','Color','r');
plot(source(1,:),source(2,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off
```

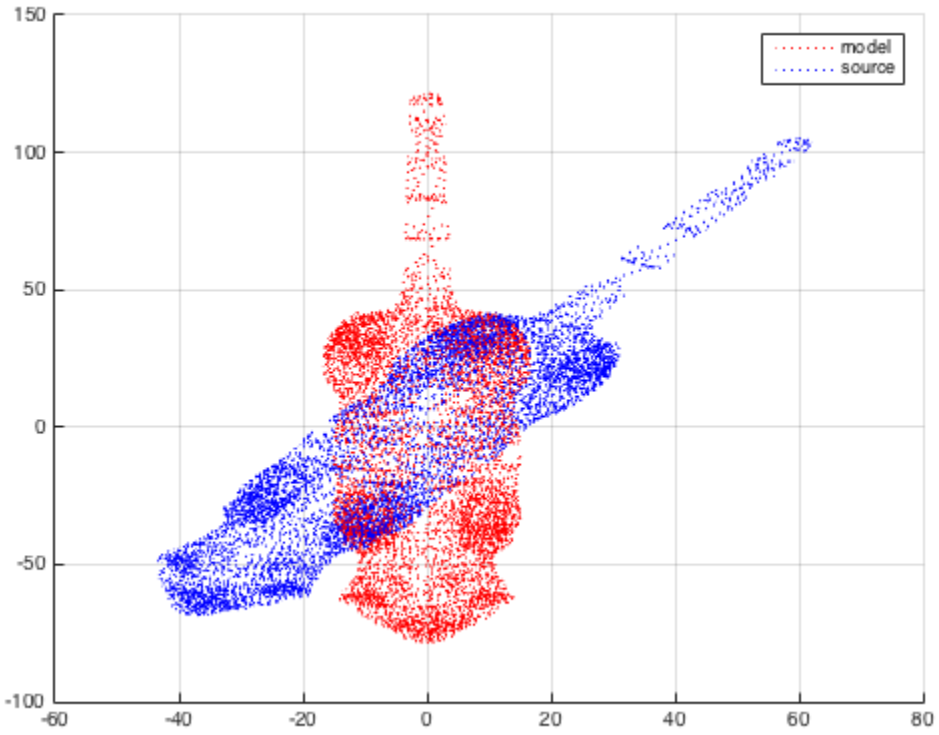


Published with MATLAB® R2015a

Table of Contents

PROBLEM 3: The ICP algorithm for 3D DATA WITHOUT NOISE

```
clear all;
close
% Model data and source data assigning for 3d  data
a = load('3D_Cat.mat');
mod = a.model;
source = a.source;
figure(1)
hold on
grid on
plot3(mod(1,:),mod(2,:),mod(3,:),'LineStyle', ':','Color','r');
plot3(source(1,:),source(2,:),source(3,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off
```

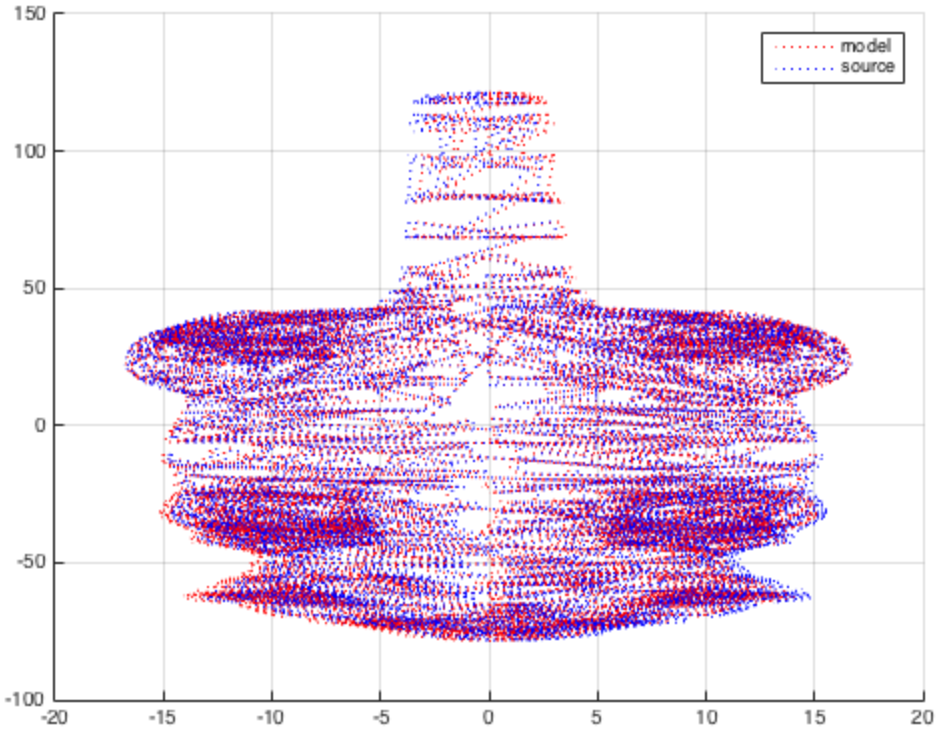


k iteration for the algorithm - ICP

```
for k = 1:1:7
[m,n] = size(mod);
[ms,ns] = size(source);
v = zeros(1,ns);
diff = zeros(1,ns);
% closest point algorithm
for j = 1:1:n
        mval = 9e99;
        val =sqrt(sum((source - repmat(mod(:,j),1,ns)).^2));
        if val<=mval
            [minim,v(j)] = min(val);
        end
end
 modchanged = mod(:,v);
% application of Principal component analysis for finding the rotation
% matrix
 centroidmod = mean(modchanged,2);
centroidsource = mean(source,2);
%Cov(x) = E(xy) - 3*E(x)*E(y)
cov = source* modchanged' - 3*centroidsource*centroidmod';
[U,~,V]=svd(cov);%calculating the SVD
Ri=V*U';%the rotation matrix
T = centroidmod - Ri*centroidsource;% Translation vlaue
Changedpossource = Ri*source + repmat(T,1,ns);%Changing the source
 data
source = Changedpossource;
end
```

plotting the data

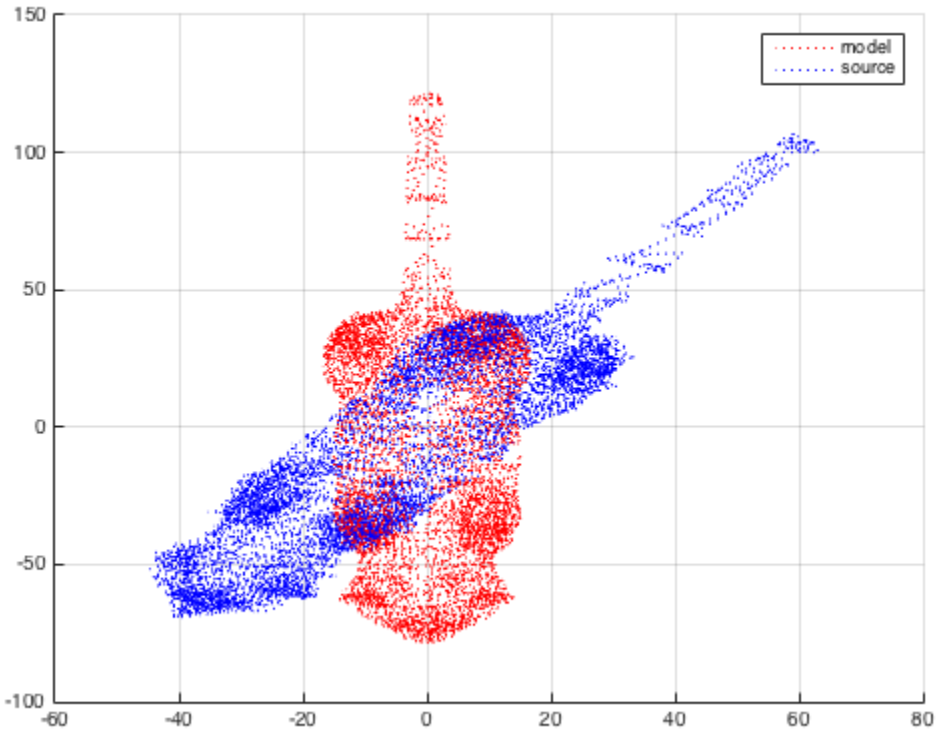
```
figure(2)
hold on
grid on
plot3(mod(1,:),mod(2,:),mod(3,:),'LineStyle', ':','Color','r');
plot3(source(1,:),source(2,:),source(3,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off


%covariance = cov(modchanged,sourcechanged);
```



Published with MATLAB® R2015a

PROBLEM 4: The ICP algorithm for 3D DATA WITH NOISE

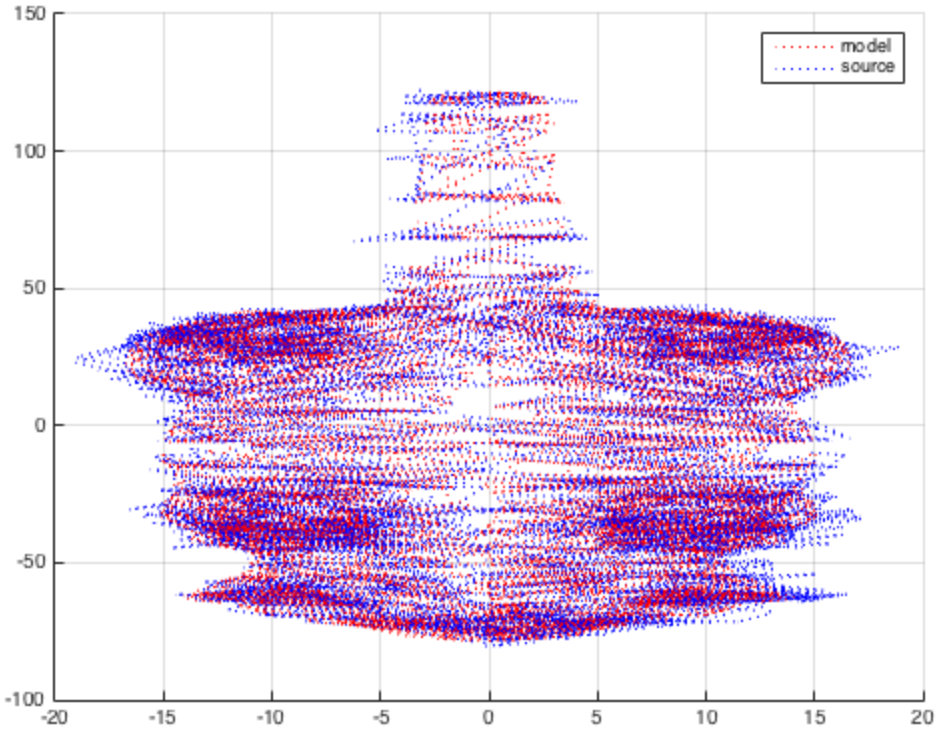
```
clear all;
close
%Model data and source data assigning for 3d Noise data
a = load('3D_Cat_Noise.mat');
mod = a.model;
source = a.source;
figure(1);
hold on
grid on
plot3(mod(1,:),mod(2,:),mod(3,:),'LineStyle', ':','Color','r');
plot3(source(1,:),source(2,:),source(3,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off
```



7 iteration for the algorithm - ICP

```
for k = 1:1:7
[m,n] = size(mod);
[ms,ns] = size(source);
v = zeros(1,ns);
```

```
diff = zeros(1,ns);
% closest point algorithm
for j = 1:1:n
        mval = 9e99;
        val =sqrt(sum((source - repmat(mod(:,j),1,ns)).^2));
        if val<=mval
            [minim,v(j)] = min(val);
        end
end
 modchanged = mod(:,v);
% application of Principal component analysis for finding the rotation
% matrix
 centroidmod = mean(modchanged,2);
centroidsource = mean(source,2);
%Cov(x) = E(xy) - 3*E(x)*E(y)
cov = source* modchanged' - 3*centroidsource*centroidmod';
[U,~,V]=svd(cov);
Ri=V*U';% rotation matrix
T = centroidmod - Ri*centroidsource;%calculating translation matrix
Changedpossource = Ri*source + repmat(T,1,ns);%changing the source
 vaulue
source = Changedpossource;
end
% plotting the data
figure(2)
hold on
grid on
plot3(mod(1,:),mod(2,:),mod(3,:),'LineStyle', ':','Color','r');
plot3(source(1,:),source(2,:),source(3,:),'LineStyle', ':','Color','b');
legend('model','source');
hold off
```



Published with MATLAB® R2015a