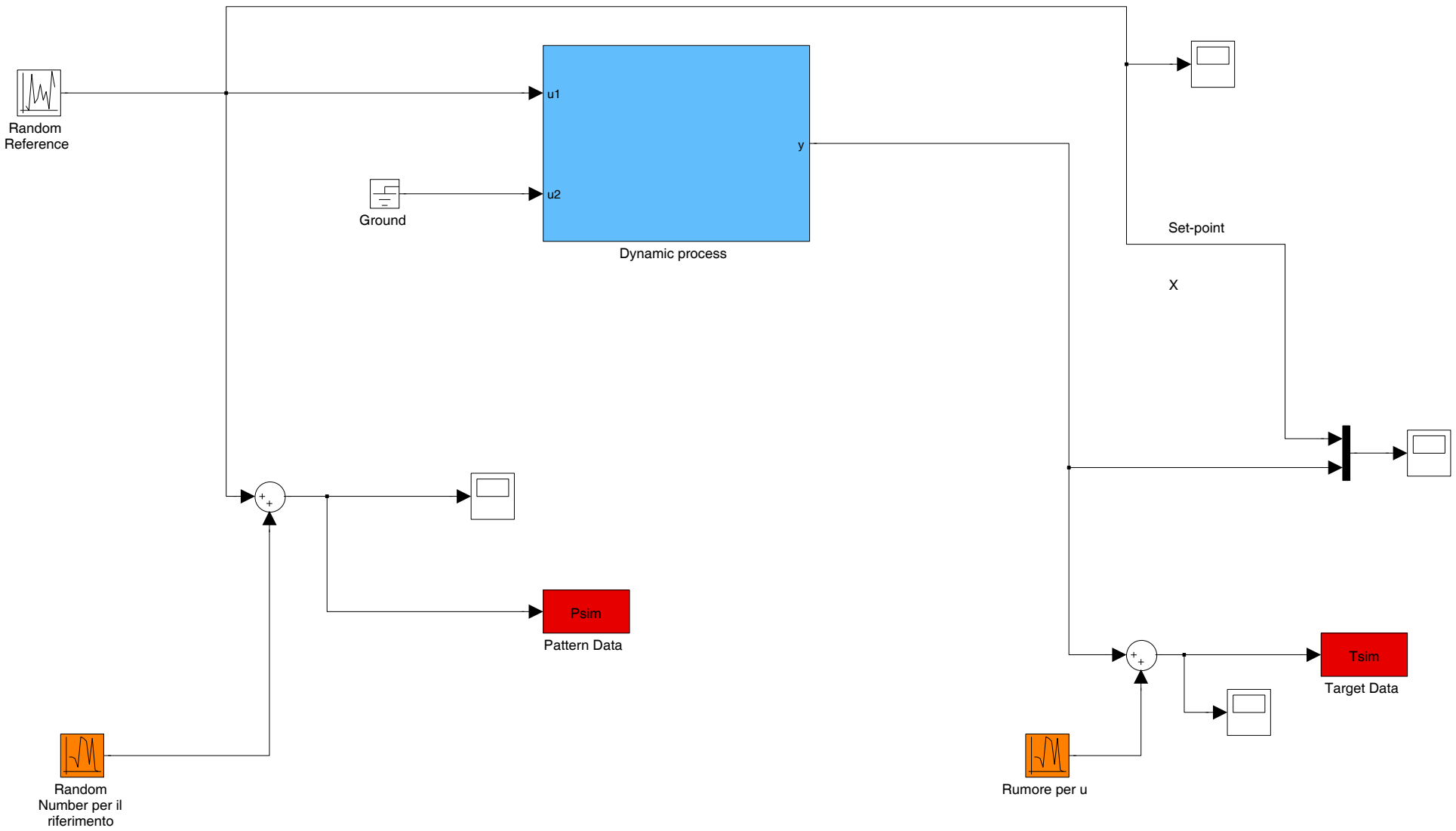
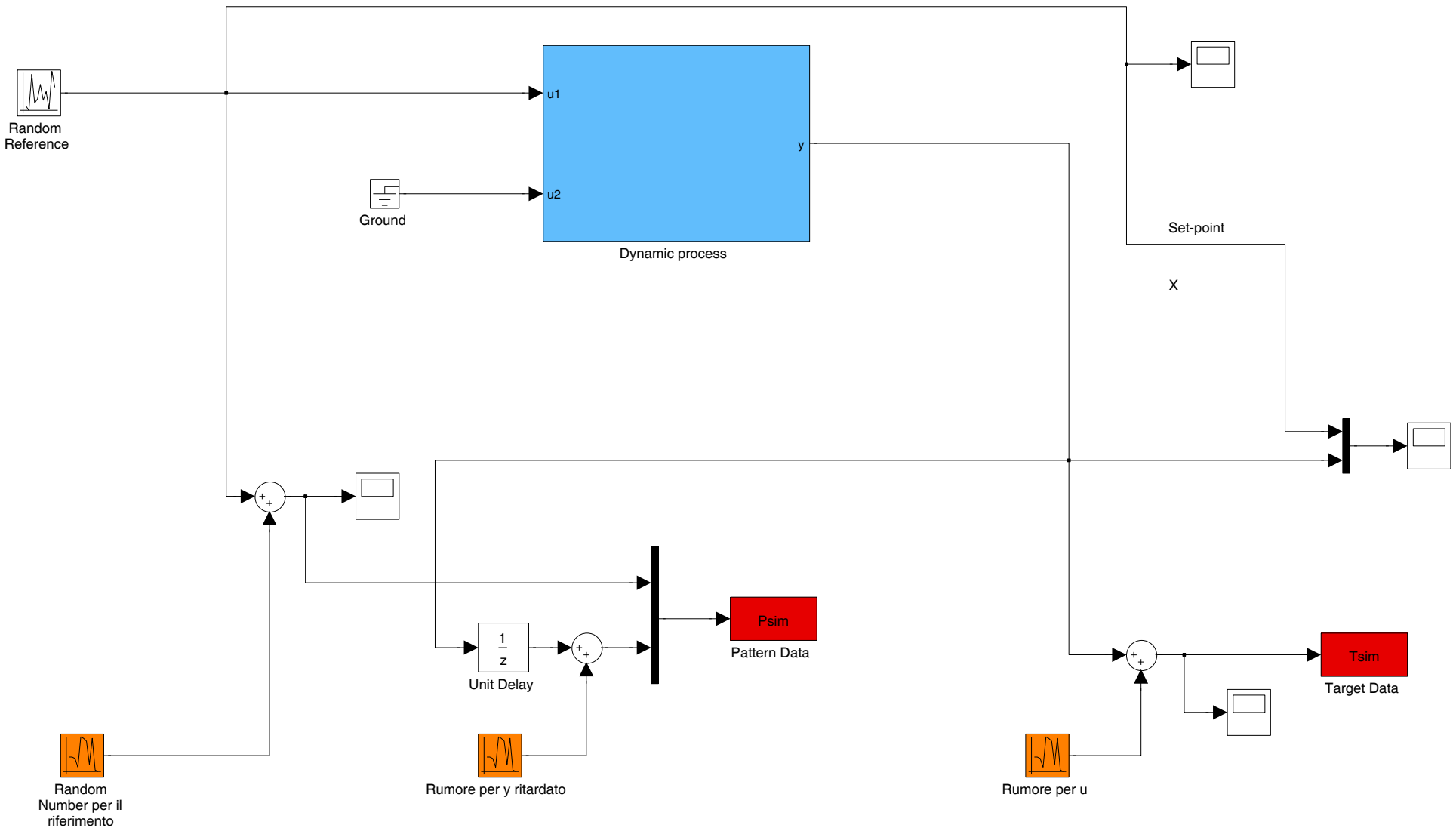
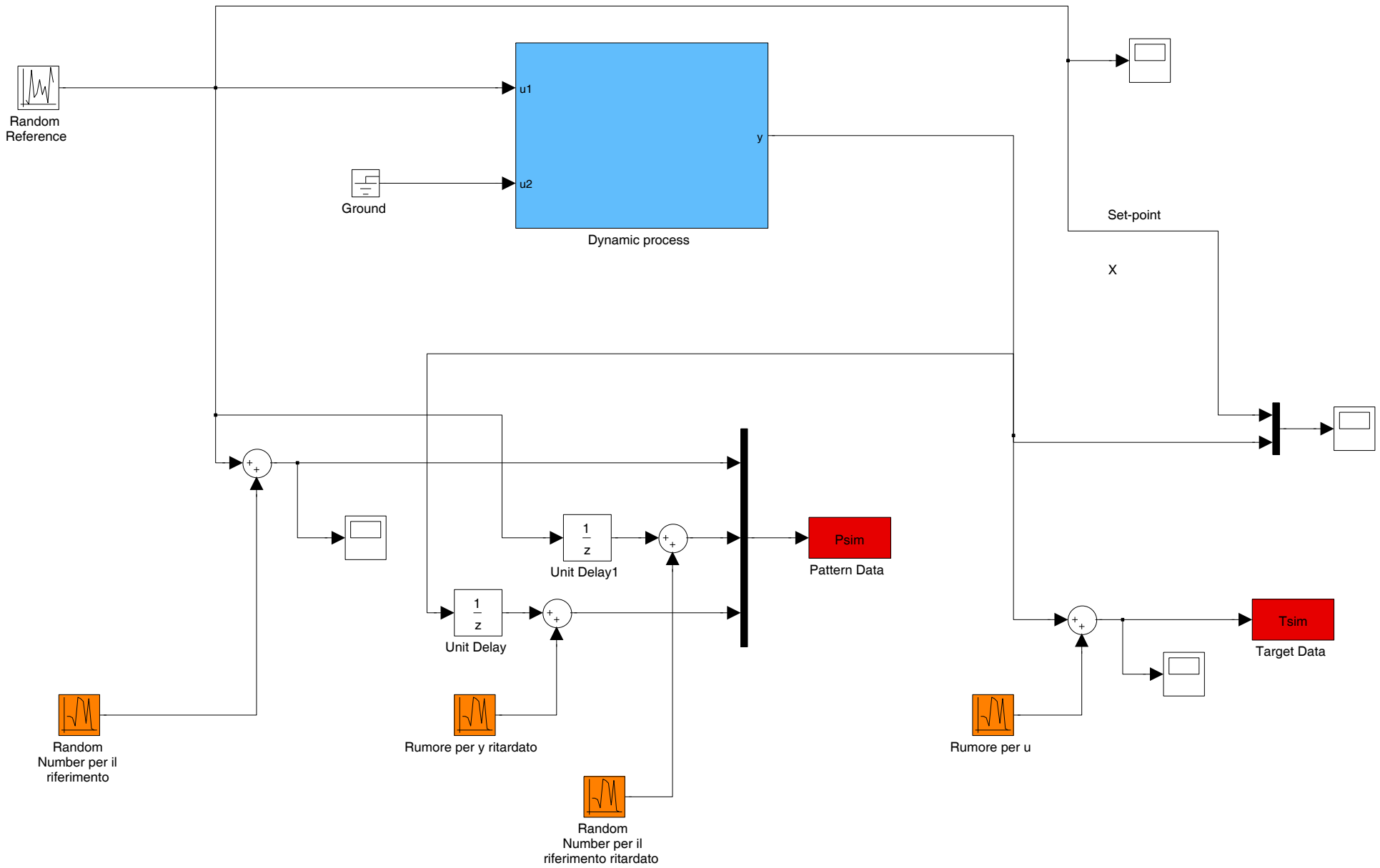
```
%%%
%%% File for the initialisation of the model parameters
%%%

M = 1;
m = 0.1;
l = 1;
g = 9.8;

x0 = 0.2;
theta0 = 0.55;

K = [-3.1623   -4.8648  -45.9014  -14.6871];
```

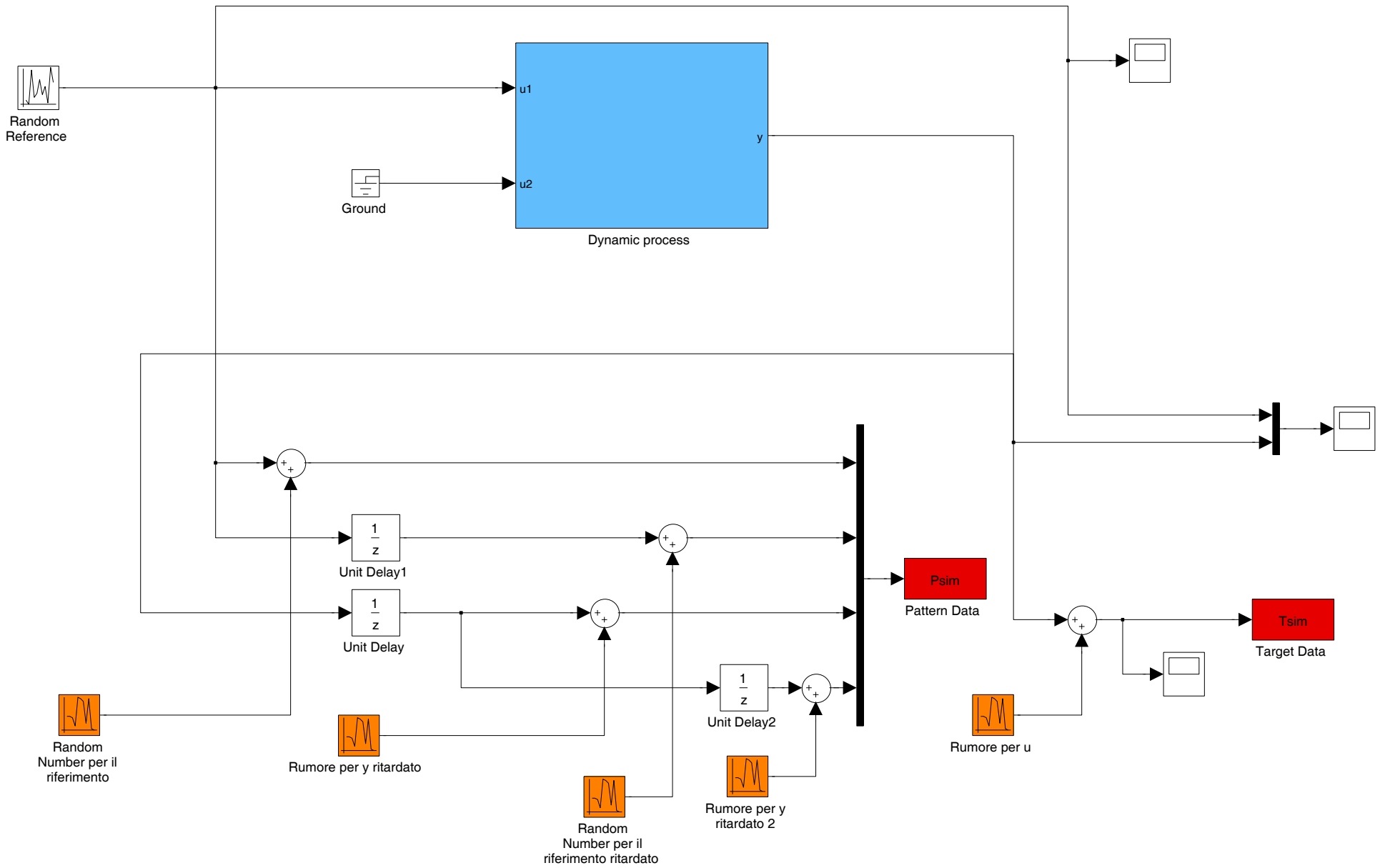
```
%%%
%%% File "train_net.m": neural network training and generation
%%%

% Caricare P e T;

P = Psim(1:round(size(Psim,1)/3),:)';
T = Tsim(1:round(size(Psim,1)/3),1)';

%%%
%%% Neural network parameters
%%%

Si = 4;  % Number of neurons in the input layer
Sh = 8;  % Number of neurons in the hidden layer
So = 1;  % Number of neurons in the output layer
         % It is equal to the rows of the matrix T

TFi = 'tansig'; % Sigmoidal tangent activation function
TFh = 'tansig';
TFo = 'purelin'; % Linear activation function

%BTF = 'traingdx'; % Function for the training of the
                   % backpropagation NN, default
BTF = 'trainlm';   % Levenberg-Marquardt backpropagation

BLF = 'learngdm';  % Backpropagation function
                   % weight/bias, default

PF  = 'mse';      % Performance function Mean Square Error
                  % default

PR  = minmax(P);  % Equal to: [min(P')' , max(P')'], it
                  % determines minimal and maximal values of
                  % inputs and output

val.P  = Psim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)';
                                                  % validation data
val.T  = Tsim(round(size(Psim,1)/3)+1:2*round(size(Psim,1)/3),:)';
test.P = Psim(2*round(size(Psim,1)/3)+1:end,:)'; % test data
test.T = Tsim(2*round(size(Psim,1)/3)+1:end,:)';

%net = newff(P,T,[Si Sh So],{TFi TFh TFo},BTF,BLF,PF);
                                          % Note: it generates a NN
```

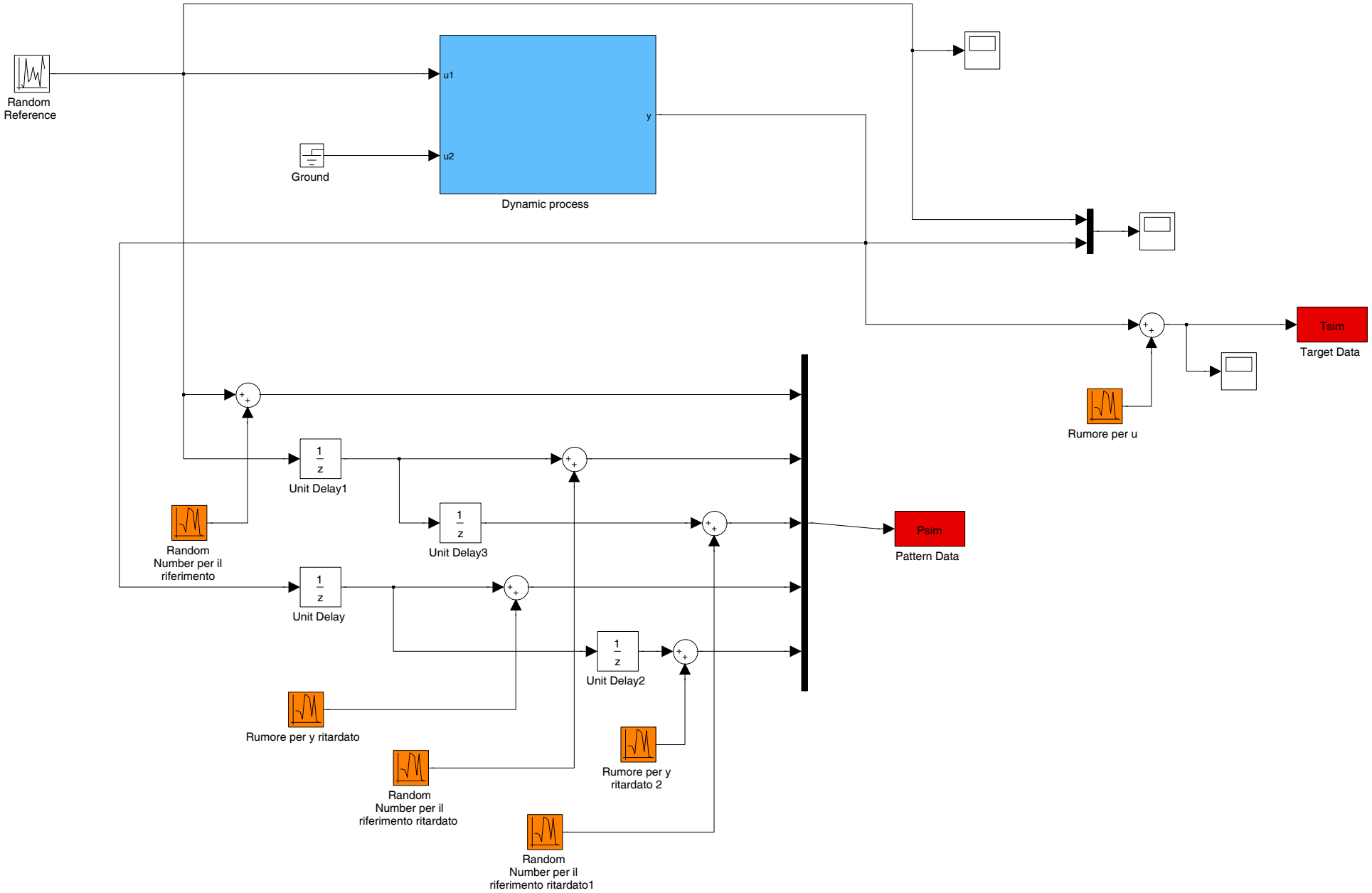
```
                                                % with 4 layers!!!
net = newff(P,T,[Si Sh],{TFi TFh TFo},BTF,BLF,PF);

%%%
%%% Parameters for the NN training
%%%

net.trainParam.epochs = 300;    % Number of epocs
net.trainParam.goal   = 1e-4;   % Value of the final error
net.trainParam.show   = 1;      % Show the plot after 1 epoch
net.trainParam.lr     = 0.05;   % Learning rate for trainlm function
net.trainParam.mc     = 0.9;    % Momentum constant: gradient value
                                % during the training phase: if 0 ->
                                % weights are changed only on the basis
                                % of the gradient; if 1 -> gradient
                                % function is completely neglected

net = train(net,P,T,[],[],val,test); % training function

Ts = 0.05;   % Sampling time
             % NOTE: it should be equal to the sampling time used
             % for collecting the matrices Tsim and Psim!!!

net.sampleTime = Ts;

gensim(net,Ts); % It creates the neural network in Simulink

return
```

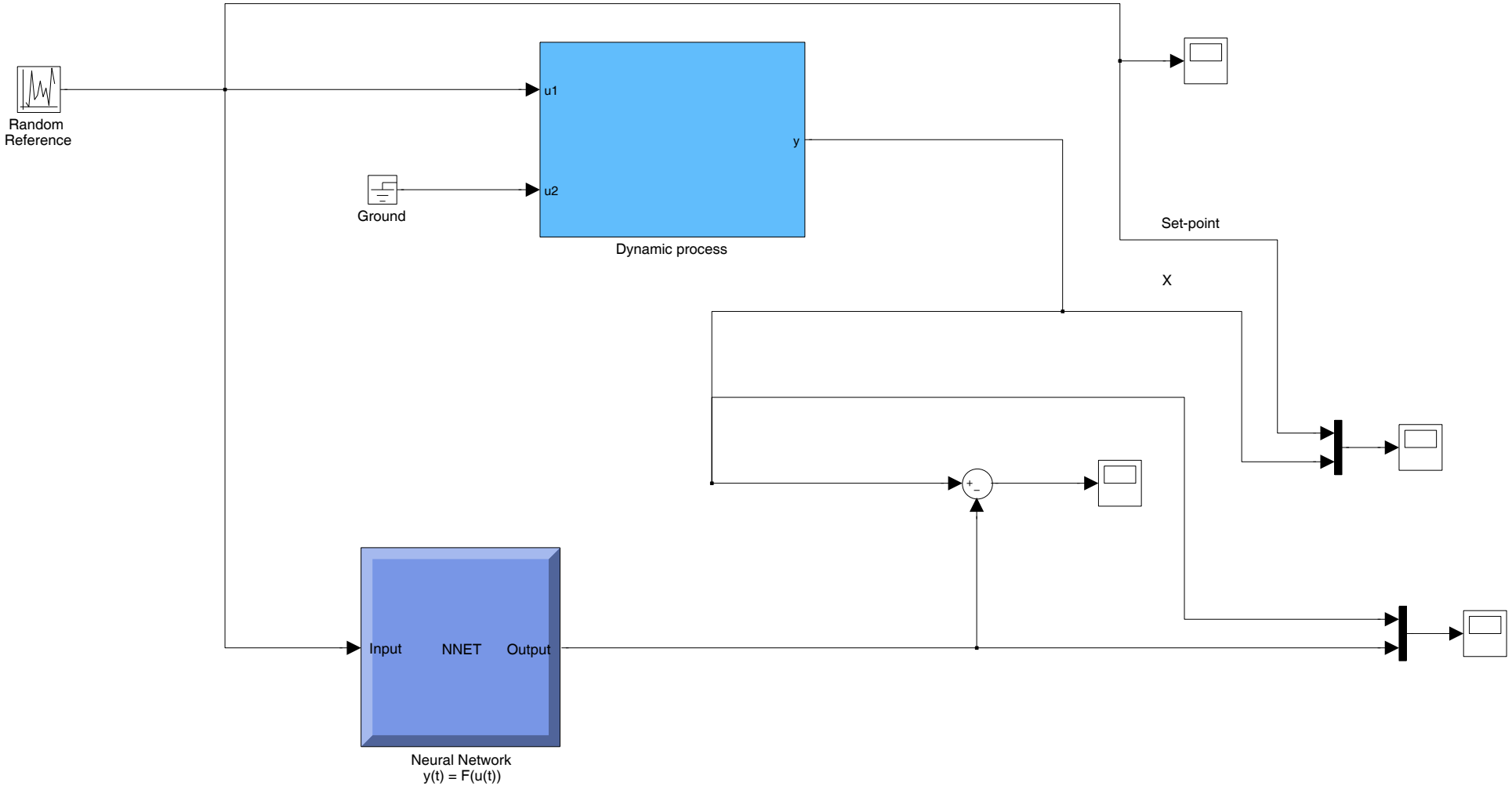

Random
Reference
u1
u2
y
Ground
Dynamic process
Set-point
X
Psim
Pattern Data
Tsim
Target Data
Random
Number per il
riferimento
Rumore per u

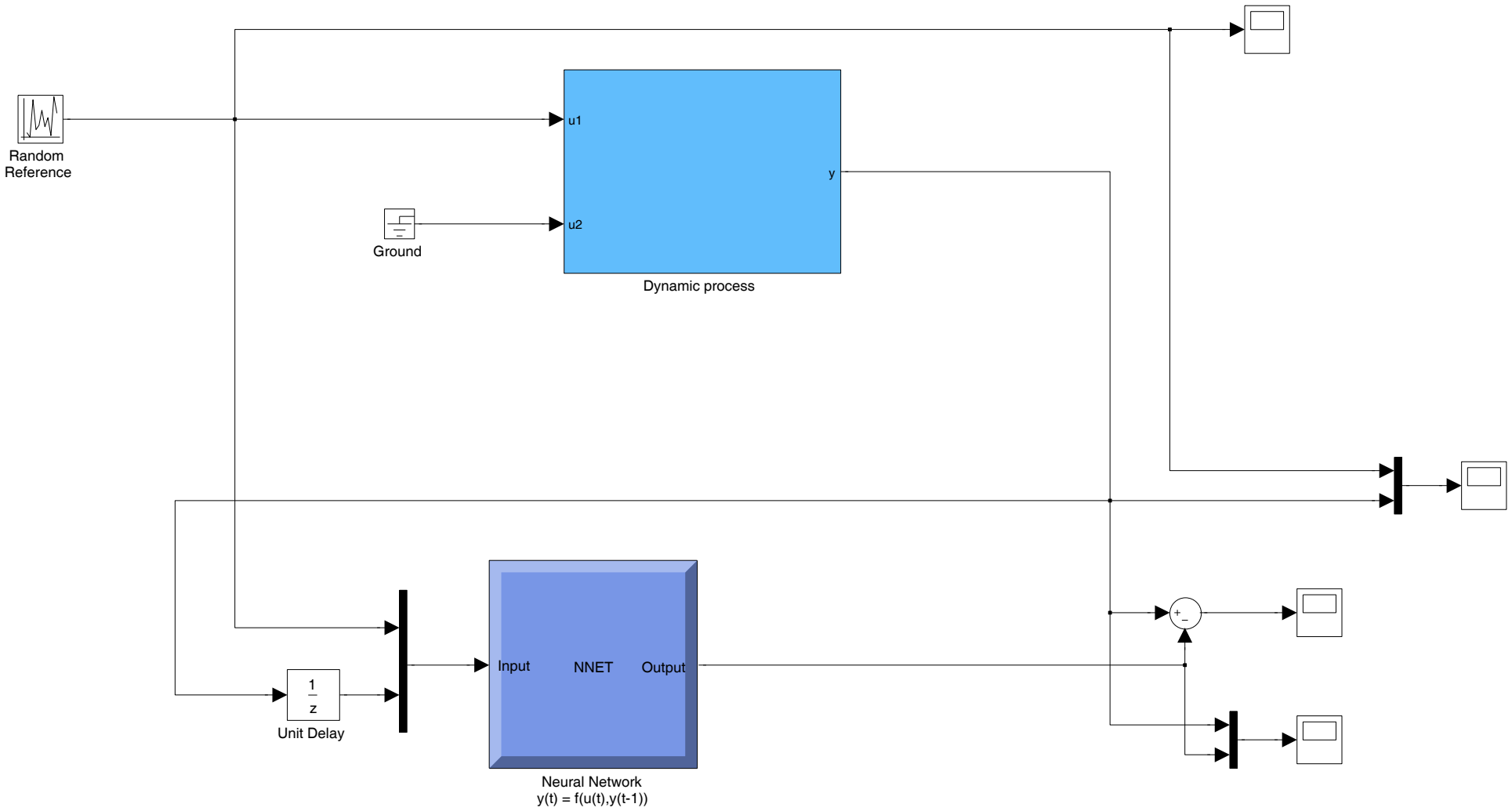

Random
Reference
u1
u2
y
Ground
Dynamic process
Set-point
X
1
z
Unit Delay
Psim
Pattern Data
Tsim
Target Data
Random
Number per il
riferimento
Rumore per y ritardato
Rumore per u

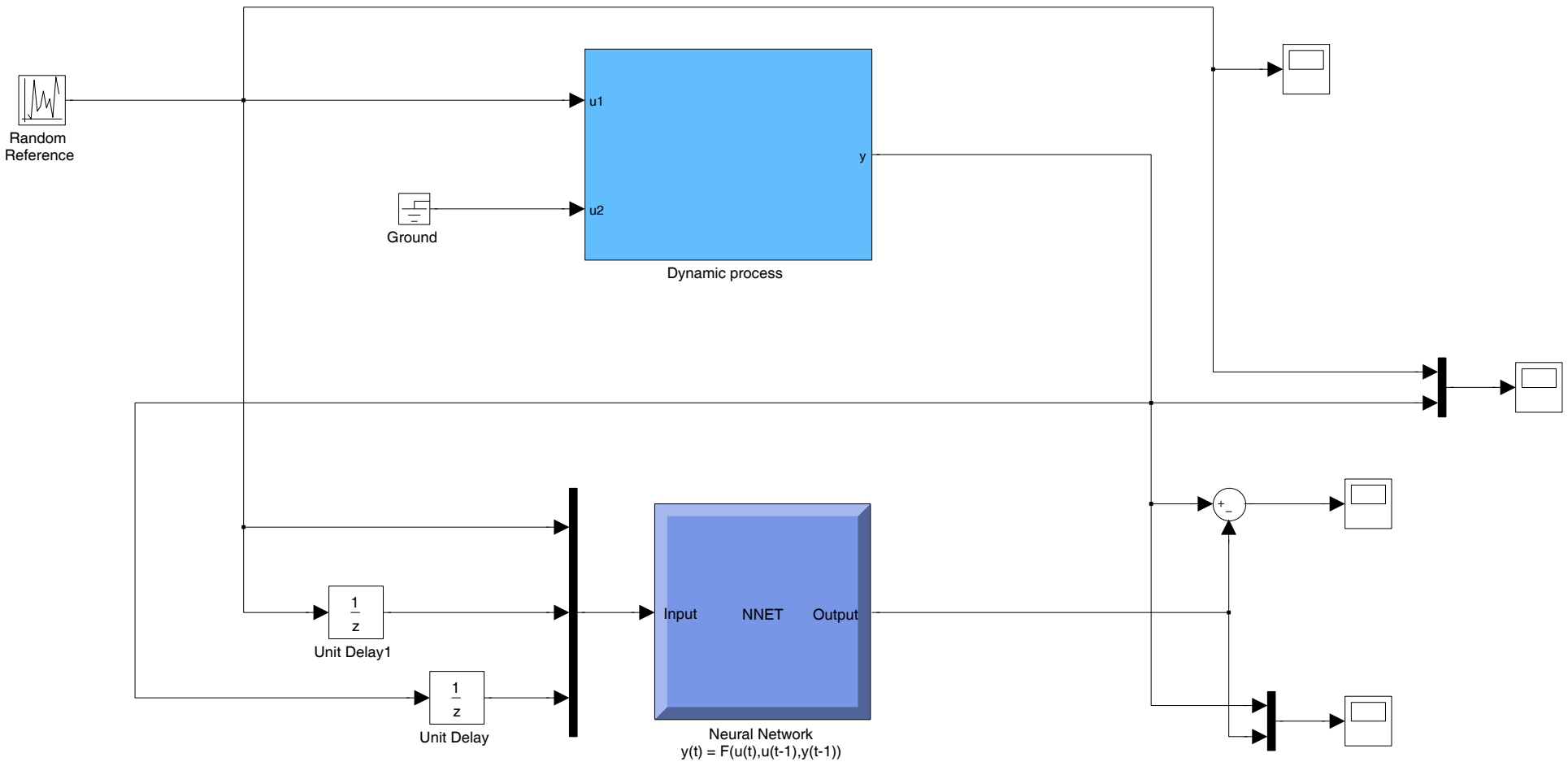

Random
Reference
u1
u2
y
Ground
Dynamic process
Set-point
X
1
z
Unit Delay1
Psim
Pattern Data
1
z
Unit Delay
Tsim
Target Data
Random
Number per il
riferimento
Rumore per y ritardato
Rumore per u
Random
Number per il
riferimento ritardato

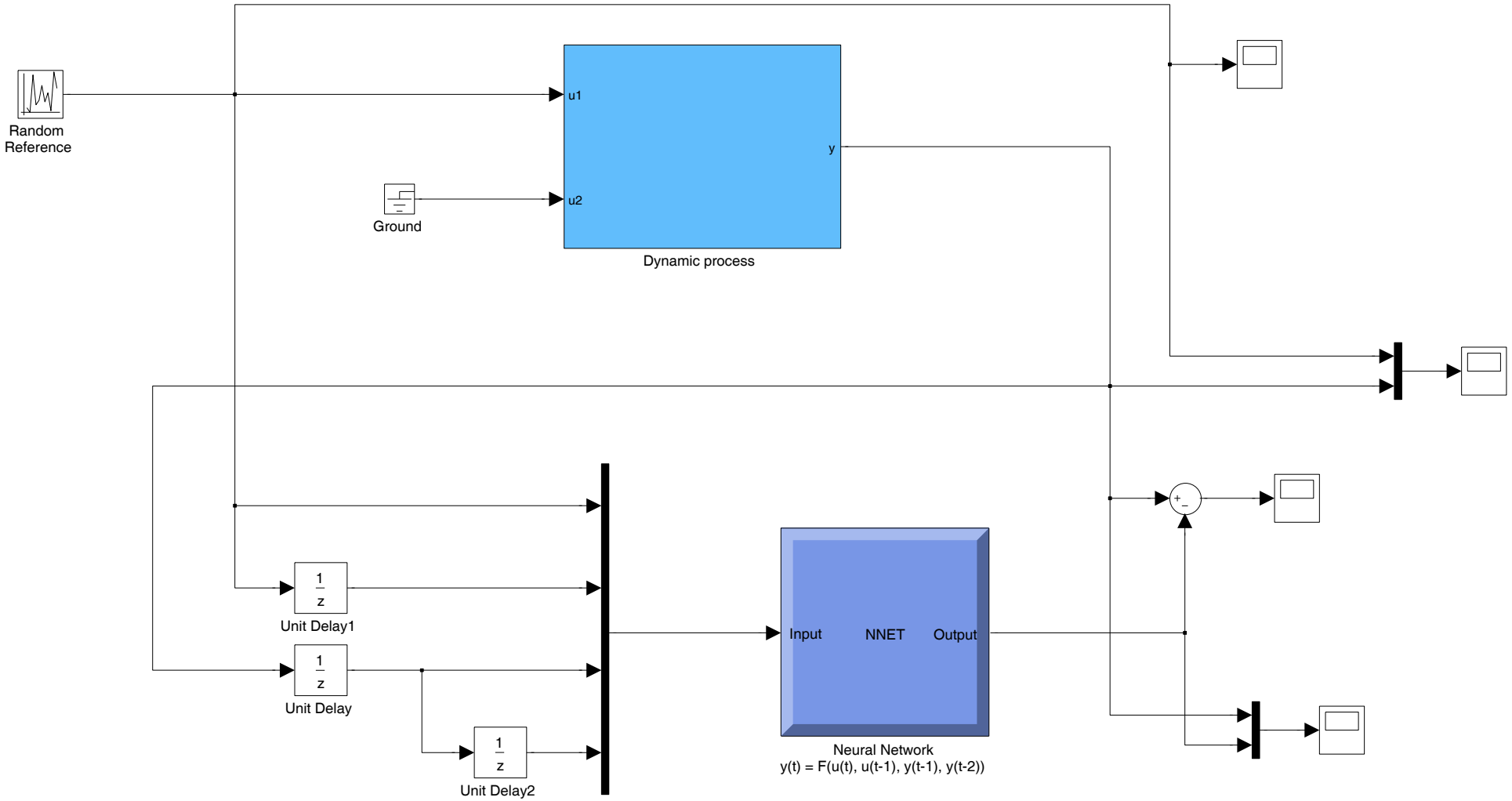

Random
Reference
u1
u2
y
Ground
Dynamic process
1
z
Unit Delay1
1
z
Unit Delay
1
z
Unit Delay2
Psim
Pattern Data
Tsim
Target Data
Random
Number per il
riferimento
Rumore per y ritardato
Random
Number per il
riferimento ritardato
Rumore per y
ritardato 2
Rumore per u

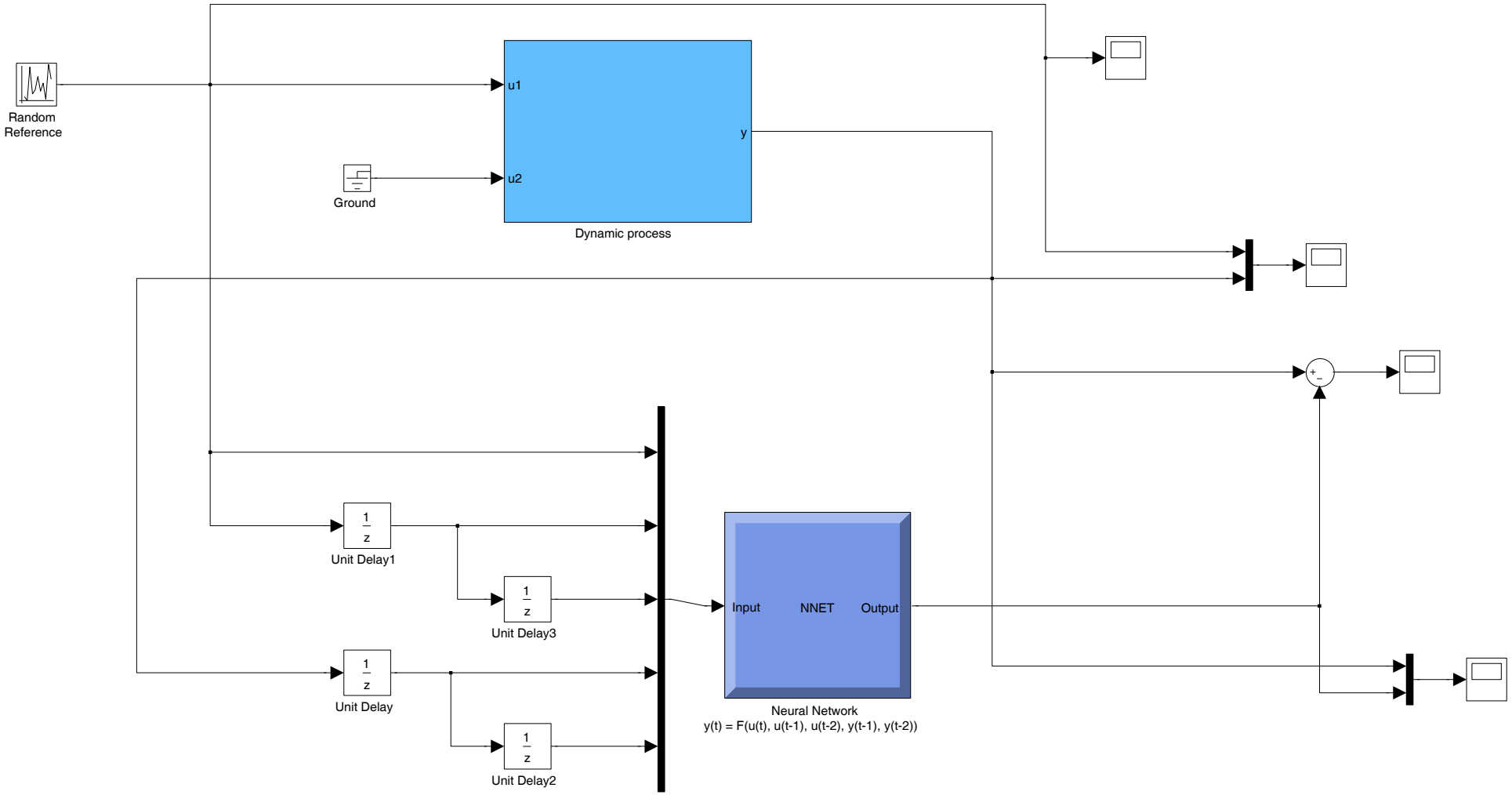

Random
Reference
u1
u2
y
Ground
Dynamic process
Tsim
Target Data
Rumore per u
1
z
Unit Delay1
Unit Delay3
Unit Delay
Unit Delay2
Random
Number per il
riferimento
Psim
Pattern Data
Rumore per y ritardato
Random
Number per il
riferimento ritardato
Rumore per y
ritardato 2
Random
Number per il
riferimento ritardato1

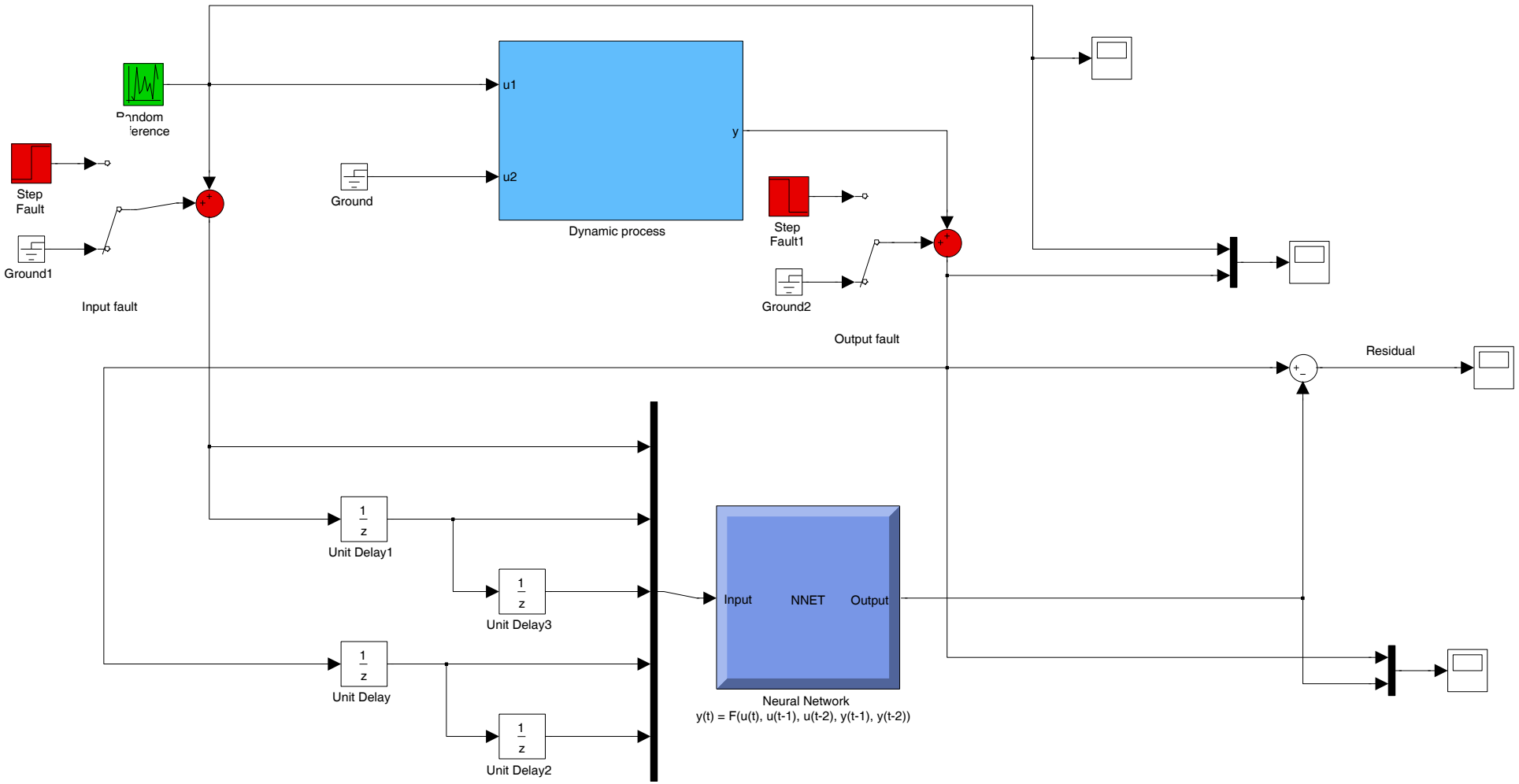

Random
Reference
u1
u2
y
Ground
Dynamic process
Set-point
X
Input
NNET
Output
Neural Network
y(t) = F(u(t))


Random
Reference
u1
u2
y
Ground
Dynamic process
1
z
Unit Delay
Input
NNET
Output
Neural Network
y(t) = f(u(t),y(t-1))


Random
Reference
u1
u2
y
Ground
Dynamic process
1
z
Unit Delay1
1
z
Unit Delay
Input
NNET
Output
Neural Network
y(t) = F(u(t),u(t-1),y(t-1))


Random
Reference
u1
u2
y
Dynamic process
Ground
1
z
Unit Delay1
1
z
Unit Delay
1
z
Unit Delay2
Input
NNET
Output
Neural Network
y(t) = F(u(t), u(t-1), y(t-1), y(t-2))


Random
Reference
u1
u2
y
Dynamic process
Ground
1
z
Unit Delay1
Unit Delay3
Unit Delay
Unit Delay2
Input
NNET
Output
Neural Network
y(t) = F(u(t), u(t-1), u(t-2), y(t-1), y(t-2))


Random
ference
Step
Fault
Ground1
Input fault
Ground
u1
u2
y
Dynamic process
Step
Fault1
Ground2
Output fault
Residual
1
z
Unit Delay1
Unit Delay3
Unit Delay
Unit Delay2
Input
NNET
Output
Neural Network
y(t) = F(u(t), u(t-1), u(t-2), y(t-1), y(t-2))