Tutorial T1: Matlab Script

```
clear all
close all

lambda = 398;                                % copper thermal conductivity
roc = 8940*385;                              % copper heat capacity
alpha = lambda/roc;                          % thermal diffusivity
xpos = 2e-3;                                 % sensor location from heated area
h = 20;                                      % heat exchange coefficient
d = 5e-3;                                    % fluxmeter diameter
Sec = pi*d^2/4;                              % section
per = pi*d;                                  % circumference
dt = 0.001;                                  % sampling time
tend = 2;                                    % experiment duration

phi0 = 5;                                    % laser heat flux
emiss = 0.75;                                % surface emissivity
phid = phi0*emiss/Sec;                       % absorbed heat flux density

time = [dt:dt:tend]';                        % time vector
freq = [1/tend:1/tend:1/dt];                 % frequency vector
N = length(time);                            % time vector dimension
omega = 2*pi*freq;                           % radial frequency
p = sqrt(-1)*omega;                          % Laplace variable
hf = model(p,lambda,alpha,h,Sec,per,xpos);   % transfer function
sysf = FRD(hf,freq);                         % conversion to system
hm = invlap('model',time,[],[],lambda,alpha,h,Sec,per,xpos); % impulse
response
idfig = 1;
figure(idfig), plot(time,hm),xlabel('time (sec)'),ylabel('impulse response')

U_1 = IDINPUT(N,'RBS',[0 0.05],[0 1]);       % RBS heat flux
U_1 = U_1*phid;                              % real absorbed heat flux density
save U_1 U_1 -ascii -double
U_1v = IDINPUT(N,'RBS',[0 0.05],[0 1]);      % RBS heat flux for validation
U_1v = U_1v*phid/2;                          % real absorbed heat flux density
for validation (linearity test also)
save U_1v U_1v -ascii -double
Tc_1 = conv(hm,U_1)*dt;                      % convolution between the heat
flux and the impulse response
Tc_1 = Tc_1(1:length(time));                 % temperature at the sensor
sig_noise = 0.05;                            % noise magnitude
Tc_1N = Tc_1+sig_noise*randn(size(Tc_1));    % noise signal
Tc_1v = conv(hm,U_1v)*dt;                    % convolution between the heat
flux and the impulse response for validation
Tc_1v = Tc_1v(1:length(time));               % temperature at the sensor for
validation
sig_noise = 0.05;                            % noise magnitude for validation
Tc_1Nv = Tc_1v+sig_noise*randn(size(Tc_1v));% noise signal for validation
idfig = idfig+1;
figure(idfig), plot(time,U_1/max(U_1),time, Tc_1/max(Tc_1),time,
Tc_1N/max(Tc_1N));
idfig = idfig+1;
figure(idfig), plot(time,U_1/max(U_1),time,
Tc_1N/max(Tc_1N),time,U_1v/max(U_1v),time, Tc_1Nv/max(Tc_1Nv));
time_id = time(1:end);                       % identification time domain
y_id = Tc_1N(1:end);                         % output data for identification
```

```
u_id = U_1(1:end);                          % input data for identification
DAT = IDDATA(y_id,u_id,dt);                 % data object

% PARAMETRIC IDENTIFICATION
Ordtest = struc(1:10,1:10,1:5);             % define the best structure
V = ARXSTRUC(DAT,DAT,Ordtest);              % the optimal structure is found
ORDERS = SELSTRUC(V,0);                     % orders for the optimal structure
sys1 = ARX(DAT,ORDERS,'focus','simulation');% ARX identification (prediction
error minimization)
idfig = idfig+1;
figure(idfig),resid(sys1,DAT)               % correlation analysis for the
residuals for ARX
sys2 = OE(DAT,ORDERS);                      % OE identification (output error
minimization)
idfig = idfig+1;
figure(idfig),resid(sys2,DAT)               % correlation analysis for the
residuals for OE
sys3 = IV4(DAT,ORDERS);                     % instrumental variable
identification
idfig = idfig+1;
figure(idfig),resid(sys3,DAT)               % correlation analysis for the
residuals for IV
present(sys1);                              % ARX model presentation
present(sys2);                              % OE model presentation
present(sys3);                              % IV model presentation
ysim1 = sim(sys1,u_id);                     % simulation using ARX solution
ysim2 = sim(sys2,u_id);                     % simulation using OE solution
ysim3 = sim(sys3,u_id);                     % simulation using IV solution
% NISI
n_o=[-1 -0.5                                % numerator orders
    0 0];                                   % numerator coefficients
d_o=[-1 -0.5 0                              % denominator orders
    0 0 1];                                 % denominator coefficients
[n_ord,num,d_ord,den,p,ecn,ecd] =
NI_SID_SVF_ident_rec(u_id,y_id,time_id,n_o,d_o,'ff',1);
% simulation using NISI solution
[ysim4]=simu_mono(num,n_ord,den,d_ord,dt,u_id);
idfig = idfig+1;
figure(idfig),plot(time_id,y_id,time_id,ysim1,time_id,ysim2,time_id,ysim3,time
_id,ysim4),xlabel('time (sec)'),...
    ylabel('sensor temperature
(°C)'),legend('measure','ARX','OE','IV','NISI'),title('simulation')
ysim1v = sim(sys1,U_1v);                    % simulation using ARX solution
ysim2v = sim(sys2,U_1v);                    % simulation using OE solution
ysim3v = sim(sys3,U_1v);                    % simulation using IV solution
                                            % simulation using IV solution
ysim4v = simu_mono(num,n_ord,den,d_ord,dt,U_1v);
idfig = idfig+1;
figure(idfig),plot(time,Tc_1Nv,time,ysim1v,time,ysim2v,time,ysim3v,time,ysim4v
),xlabel('time (sec)'),...
    ylabel('sensor temperature
(°C)'),legend('measure','ARX','OE','IV','NISI'),title('validation')
[ir1,tir1] = impulse(sys1,tend);            % impulse response using ARX
[ir2,tir2] = impulse(sys2,tend);            % impulse response using OE
[ir3,tir3] = impulse(sys3,tend);            % impulse response using IV
uh=zeros(size(time)); uh(1:2) = 1/dt;
[ir4] = simu_mono(num,n_ord,den,d_ord,dt,uh);% impulse response using NISI
```

```
idfig = idfig+1;
figure(idfig),plot(time,hm,tir1,ir1,tir2,ir2,tir3,ir3,time,ir4),xlabel('time
(sec)'),...
    ylabel('impulse response (°C)'),legend('model','ARX','OE','IV','NISI')

%figure(8), bode(sysf,Gh,sys2,'sd',3,'fill'),legend('real TF','estimate TF
from SPA','estimate TF from IM')
% f0 = 0.1; % lower frequency
% f1 = 10; % higher grequency
% f = f0+(f0-f1)/time(end).*time; % frequency
% U_2 = sin(2*pi*f.*time); % linear swept frequency sequence
% U_2 = U_2+1; % heat flux must be positive
%
% Tc_2 = conv(hm,U_2)*dt;
% Tc_2 = Tc_2(1:length(time));
%
% figure(3), plot(time,U_2/max(U_2),time, Tc_2/max(Tc_2));
```