Numerical Optimization Project

Azad Almasov

November 6, 2020

STATEMENT OF THE PROBLEM

NOTE: Some sentences in STATEMENT OF THE PROBLEM section were not paraphrased

Given linear differential equation:

$$\frac{dx(t)}{dt} = Ax(t) + Cu \tag{1}$$

where, A and C are model parameters, u(t) is input function, x is our variable. Remember that x and u are n dimensional vectors, thus, A and C are nxn dimensional matrices. For example, as an variable you may have temperature (T), pressure (P), flow rate measurements (q) etc. Thus, these variables creates vector x.
Given the parameters of the model, which are A and C, the stimulus values, u(t), and the initial state of the system x(0), the response or solution x(t) can be determine numerically. This is the forward problem.
The inverse problem is the problem of

finding the parameters in the model from a sequence of experiments. Specially we want to determine A and C given a sequence of experiments where we input different functions u(t) and measure the output x(t). And since x depends on the parameters, let's write the solution as x(t; A;C).
when you conduct at experiment, measurements are always contaminated by noise and samples are only taken at a discrete set of point say $\{t_1, ...t_K\}$. So instead of being able to measure x directly we instead measure:

$$y^i(t_k) = x^i(t_k) + \epsilon \tag{2}$$

where we will assume that ϵ is a Gaussian random variable which is mean zero and variance $\sigma^2 I$.

A standard approach to such a problem is to try to choose A, B and C to minimize

$$f(A, C) = \sum_{i=1}^{N} \sum_{k=1}^{K} ||y^i(t_k) - x^i(t_k, A, C)||^2 \tag{3}$$

SOLUTION

Manual for Code

In this example, I have 3 variables and 3 different input functions for each experiment. Each of these input functions changes for different each experiment. Input functions are given in 'inFun' function. If you pay attention you will see that the experiemnt number i is included in those equations so that they will be different for different experiemnts. So, for my different test examples, 'inFun' will have to be modified. I used conditional statement to choose desired example by the user. So all my examples are in this script you just need to modify the example number

In function 'Measurement', I call these functions ('inFun' - input function, 'dfx' - ODE, 'eks' - numrical solution of ODE for x) inside for loop for each measurements. I also calculate measurements (yk) by adding error to each solution with mean zero and 0.1 variance. Function 'MSER' calculates mean square error.

NOTE: My different examples are choosing different input functions. However, besides these you can go and modify error and number of experiments or initial guess of model paramaters or number of measurement time etc.

Step 1: So I defined my input functions for different examples which changes from one experiment to another experiment. To do so I have used logical operations:

```
function u = inFun(tt,i,example) Arbitrary Input functions
if example == 1;
cc = sin(tt/i); Each experiment these input functions keep changing
z = cos(tt/i)-sin(tt/i); i is the ith experiment
uu = cos(tt/i);
u = [z;cc;uu];
elseif example == 2;
cc = sin(tt*i);
z = cos(tt*i)-sin(tt*i);
u = [z;cc];
else
cc = sin(tt*i)+cos(tt/i);
z = cos(tt*i)-sin(tt/i);
u = [z;cc];
end
end
```

Step 2: Using this input function I have defined another function for ODE:

```
function df = dfx(tt,x,A,C,i,example)
df = A*x + C*inFun(tt,i,example);
```

end

Step 3: Then using this ODE, I have defined another function where I solve this ODE for x numerically:

function $[t, xx]$ = *eks(A,x0,tspan,i,n,example) Solve ODE*
*CC = A(1:n,1+n:2*n);*
AA = A(1:n,1:n);
$[t, xx] = ode45(@(tt, x)dfx(tt,x,AA,CC,i,example),tspan,x0');$
end

Step 4: Then using this output variables I have created function where I solve x for different experiments. And using x and adding normal error with zero mean and 0.1 variance I creates measured data for each experiment and for each variable - y_k

function $[xxx, ms, tt]$
=Measurement(A,x0,tspan,m,n,example) It saves every experiemnt into seperate cells
rng('default');
$error = normrnd(0, 0.1, [length(tspan), 1]) * ones(1, n)$; *Normal error*
for j=1:m
$[t, sol\{j\}]$ = *eks(A,x0,tspan,j,n,example);*
$y\{j\} = sol\{j\}$ + *error; Measurement j:1,....m*
end
xxx = sol; Real model output
ms = y; Measuremnets sells for each experiment
tt = t; measured time cells for each measurement
end

Step 5: Then we use model parameters A and C, and we generate measured data for each experiment and for each variable-x. Then I created function that calculates mean square error:

function mser = MSER(A,x0,tspan,m,n,yk,example)
for i=1:m
$[x, \ , t]$ = *Measurement(A,x0,tspan,m,n,example);*
for j=1:length(t)
$asd(j) = (norm(yk\{i\}(j,:) - x\{i\}(j,:)))^2;$
end dsa(i) = sum(asd);
end
mser = sum(dsa);
end

Step 6: I have used quasi-newton method for optimization, using the Matlab routine $fminunc$ with the following options and initial guess A0:

$options = optimoptions(@fminunc,'Display','iter','Algorithm','quasi-newton','TolFun',1e-5,'TolX',1e-5,'MaxIter',1000);$

$func = @(ff)MSER(ff, x0, tspan, m, n, yk, example);$
$opt = fminunc(func, A0, options)$

As you can see in function 'eks', I get my model parameters A and C by splitting my combined model parameter into two parts. So, I used combined model parameter instead of using those model parameters separately. I do so because optimization algorithm can get one initial guess as matrix parameter. If you had multiple scalar parameters, you would have been able use vector initial guesses of those variables as input to your optimization routine: Combined parameter is defined as block matrix:
$B = [A, C]$
For initial guess I just added 0.01 t0 my original combined model parameter:
$B_0 = B + 0.01$

Step 7: After getting guessed combined model parameter I have also calculated
$err = ||B_{est} - B_0||$.

CASES

For all cases random noise is the same as well as number of experiments.
I have looked the following cases:

Case 1:
We have 3 variables. Input functions are:
$u_1 = sin(tt/i)$
$u_2 = cos(tt/i) - sin(tt/i)$
$u_3 = cos(tt/i)$

Combined model parameter to generate measured data is:

$$B = \begin{bmatrix} -0.166 & -0.120 & -0.074 & 0.087 & 0.087 & 0.040 \\ -0.288 & -0.058 & -0.138 & 0.0051 & 0.084 & 0.123 \\ -4.57e-05 & -0.037 & -0.158 & 0.11 & 0.066 & 0.06 \end{bmatrix}$$

Case 2:
We have 2 variables. Input functions are:
$u_1 = sin(tt * i)$
$u_2 = cos(tt * i) - sin(tt * i)$

Combined model parameter to generate measured data is:

$$B = \begin{bmatrix} -0.167 & -4.57e-05 & 0.087 & 0.11 \\ -0.288 & -0.121 & 0.005 & 0.087 \end{bmatrix}$$

Case 3:
We have 2 variables. Input functions are:
$u_1 = sin(tt * i) + cos(tt/i)$
$u_2 = cos(tt * i) - sin(tt/i)$

Combined model parameter to generate measured data is:
$$B = \begin{bmatrix} -0.167 & -4.57e-05 & 0.087 & 0.11 \\ -0.288 & -0.121 & 0.005 & 0.087 \end{bmatrix}$$

Then I repeated these cases for the case where I have more experiment (10 experiments).

RESULTS

Measured Data:

I have plotted measured vs time for each case for each variables. All experiments result for each variable are in the same plot (Figures 1, 2 and 3).

I have also plotted measured data for 10 experiment cases (Figures 4, 5 and 6).

Optimization Results:

Case 1, 5 Experiments Iterations: 85
Original MSE was: 35.4
Minimized MSE from Optimization is: 35
err = 0.0559

Estimated combined model parameter:
$$B_{est} = \begin{bmatrix} -0.1956 & -0.1346 & -0.0744 & 0.0929 & 0.0945 & 0.0441 \\ -0.3160 & -0.0711 & -0.1362 & 0.0116 & 0.0909 & 0.1272 \\ -0.0267 & -0.0505 & -0.1639 & 0.1154 & 0.0735 & 0.0629 \end{bmatrix}$$

Case 2, 5 Experiments Iterations: 27
Original MSE was: 23.58
Minimized MSE from Optimization is: 23.26
err = 0.13

Estimated combined model parameter:
$$B_{est} = \begin{bmatrix} -0.149 & -0.073 & 0.088 & 0.113 \\ -0.278 & -0.230 & 0.0048 & 0.0885 \end{bmatrix}$$

Case 3, 5 Experiments Iterations: 27
Original MSE was: 23.58
Minimized MSE from Optimization is: 23.29

err = 0.0505

Estimated combined model parameter:
$B_{est} = \begin{bmatrix} -0.187 & -0.027 & 0.090 & 0.116 \\ -0.312 & -0.145 & 0.008 & 0.0988 \end{bmatrix}$

Case 1, 10 Experiments Iterations: 86
Original MSE was: 70.75
Minimized MSE from Optimization is: 69.85

err = 0.0748

Estimated combined model parameter:
$B_{est} = \begin{bmatrix} -0.195 & -0.135 & -0.084 & 0.092 & 0.096 & 0.0458 \\ -0.327 & -0.079 & -0.151 & 0.0114 & 0.0959 & 0.131 \\ -0.0355 & -0.0555 & -0.171 & 0.114 & 0.077 & 0.066 \end{bmatrix}$

Case 2, 10 Experiments Iterations: 40
Original MSE was: 47.16
Minimized MSE from Optimization is: 46.55
err = 0.1334

Estimated combined model parameter:
$B_{est} = \begin{bmatrix} -0.146 & -0.073 & 0.09 & 0.113 \\ -0.273 & -0.230 & 0.0028 & 0.085 \end{bmatrix}$

Case 3, 10 Experiments Iterations: 34
Original MSE was: 47.17
Minimized MSE from Optimization is: 46.61
err = 0.0616

Estimated combined model parameter:
$B_{est} = \begin{bmatrix} -0.196 & -0.034 & 0.090 & 0.117 \\ -0.321 & -0.143 & 0.01 & 0.098 \end{bmatrix}$
When you run the script in command line you will see original combined model parameter matrix (B), original mean square error ($meansqrter$), initial guess for combined model parameter ($B0$), estimated combined model parameter matrix ($Best$), estimated mean square error and error ($mismatch$) of model parameters (err).

I plotted how estimated parameters help our function to match observed noisy data - regression result. I have plotted these results with measured noisy data as well. These plots are for 5 experiment cases (Figures 7,7 and 7).

CONCLUSIONS

Comparing the mismatch between original combined model parameter and estimated model parameter (*err*), I can say the followings:
1. Choosing appropriate input function is important. From plots 7, 8 and 9 you can see that second case resulted more noises and mismatch is high (0.13)
2. Our optimization algorithm is sensitive to noise a little bit. We can see this from the error calculations of cases. Each time our estimated mse is lower than original mse. This shows that our model tried to mask the effect of noise as well. However, difference is not that much. I can say that sensitivity of the algorithm to my noise was negligible.
3. For more experiment cases (10 experiments) our mismatch is little bit higher than that of in less experiment cases (5 experiments). I think, this is because our model become more sensitive to noise. We can see this if we compare the estimated and original mse. The reason for the model being sensitive to the noise of the data is that when we use more training data (more experiments) we may end up with overfitting to the noise.

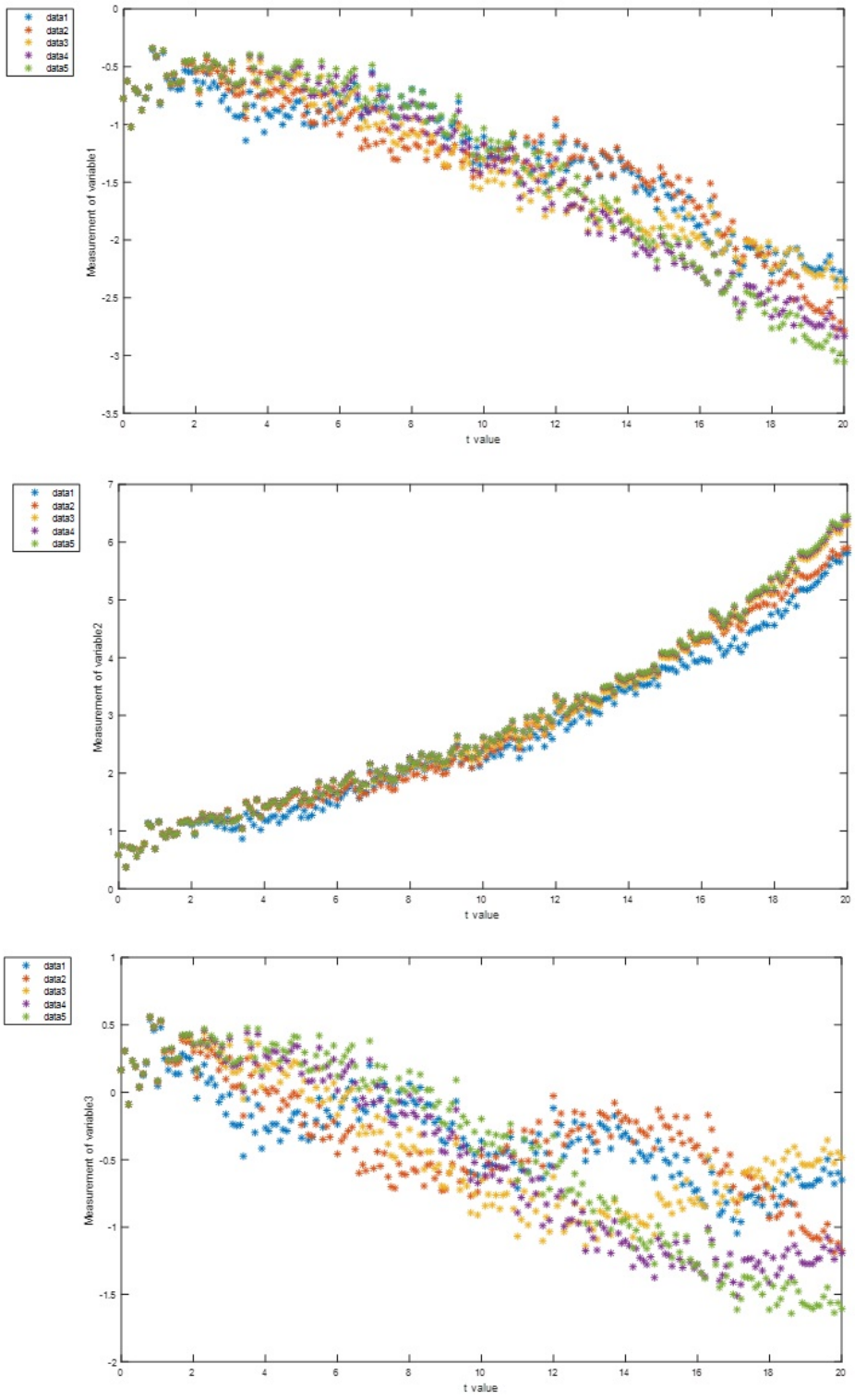

Figure 1: Measured data for Case 1, 5 experiments

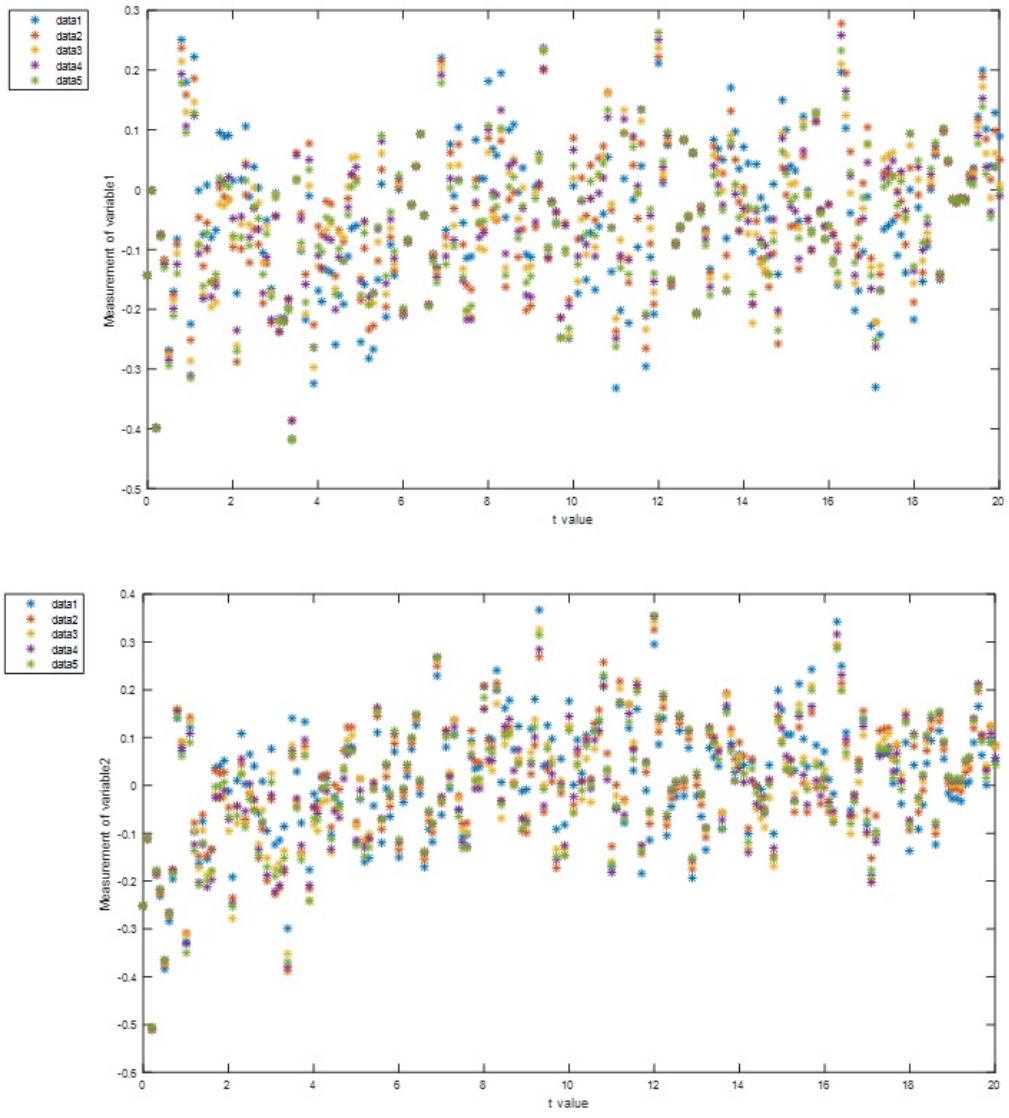

Figure 2: Measured data for Case 2, 5 experiments

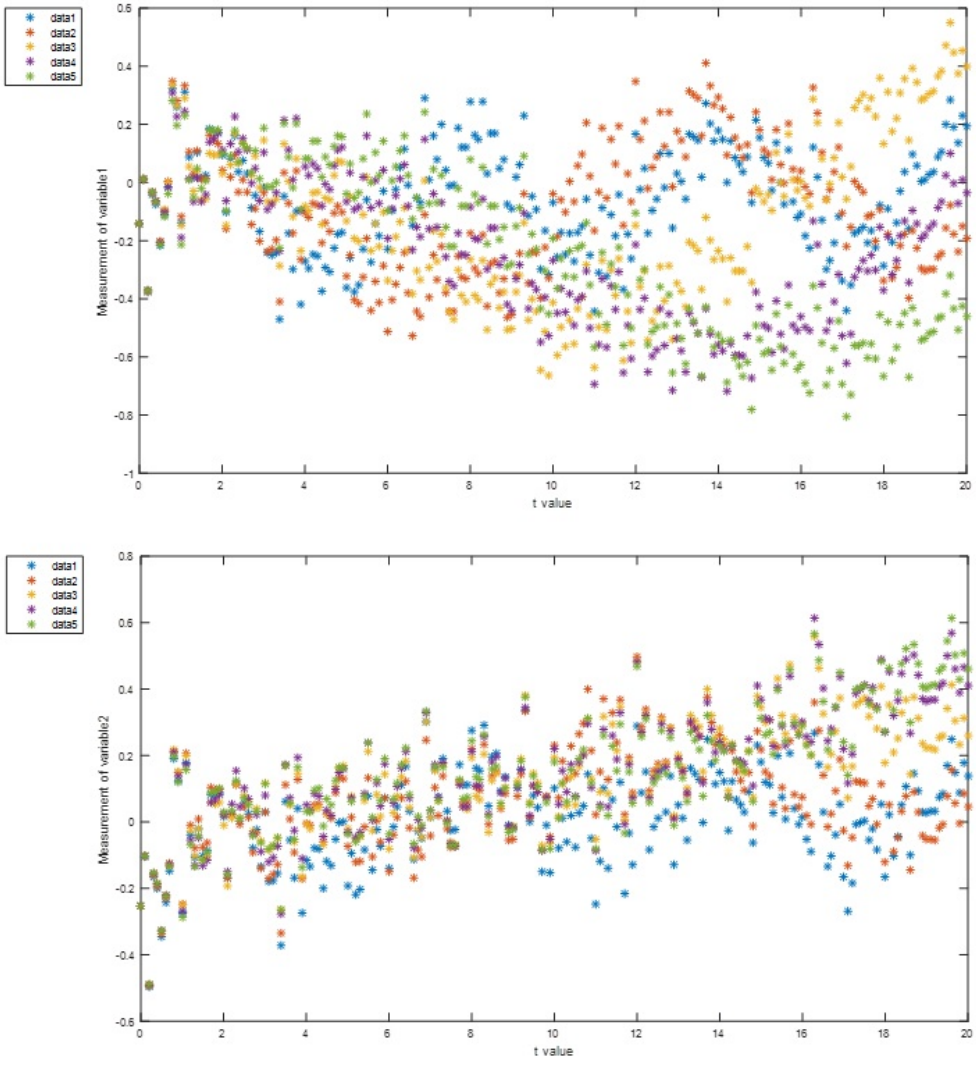

Figure 3: Measured data for Case 3, 5 experiments

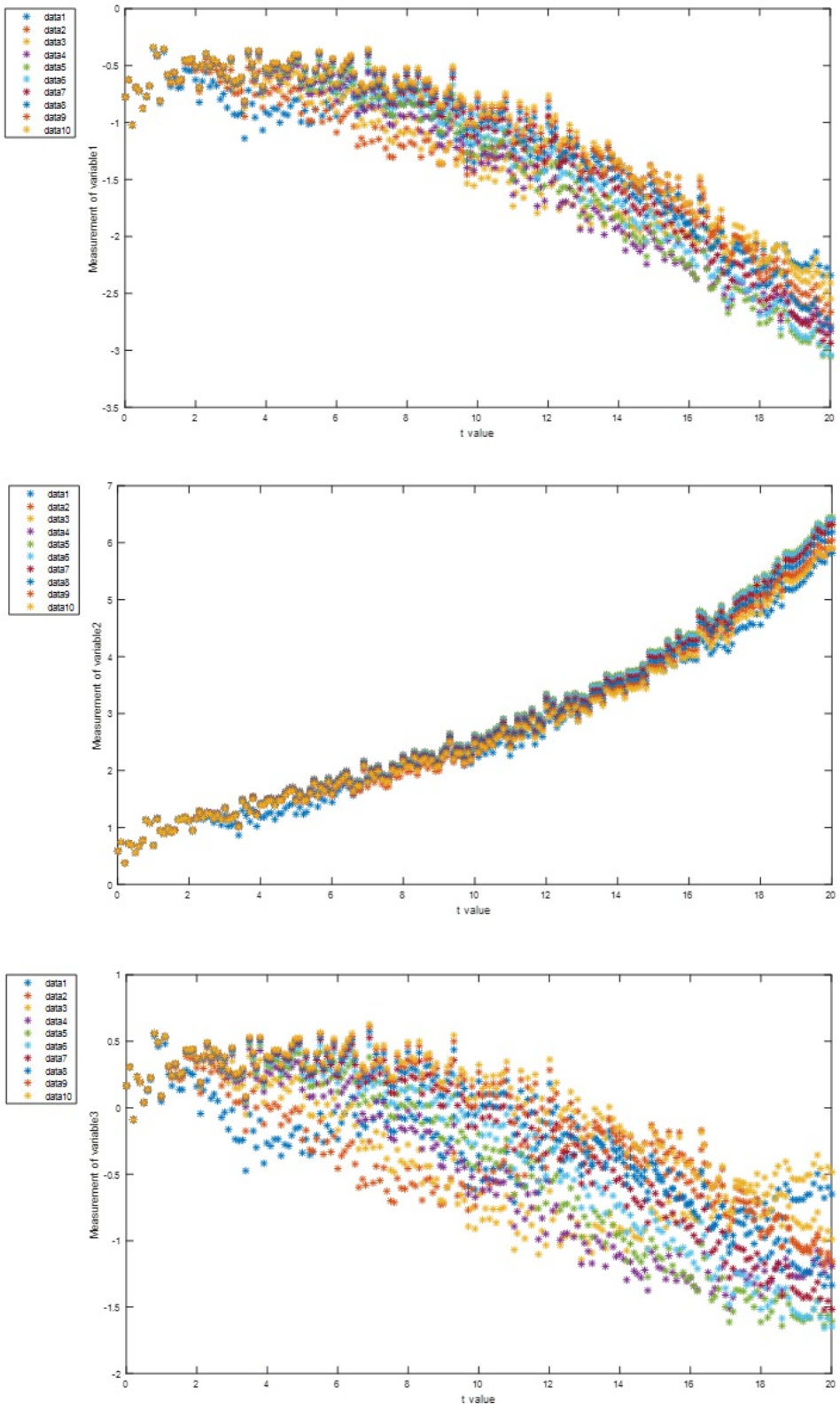

Figure 4: Measured data for Case 1, 10 experiments

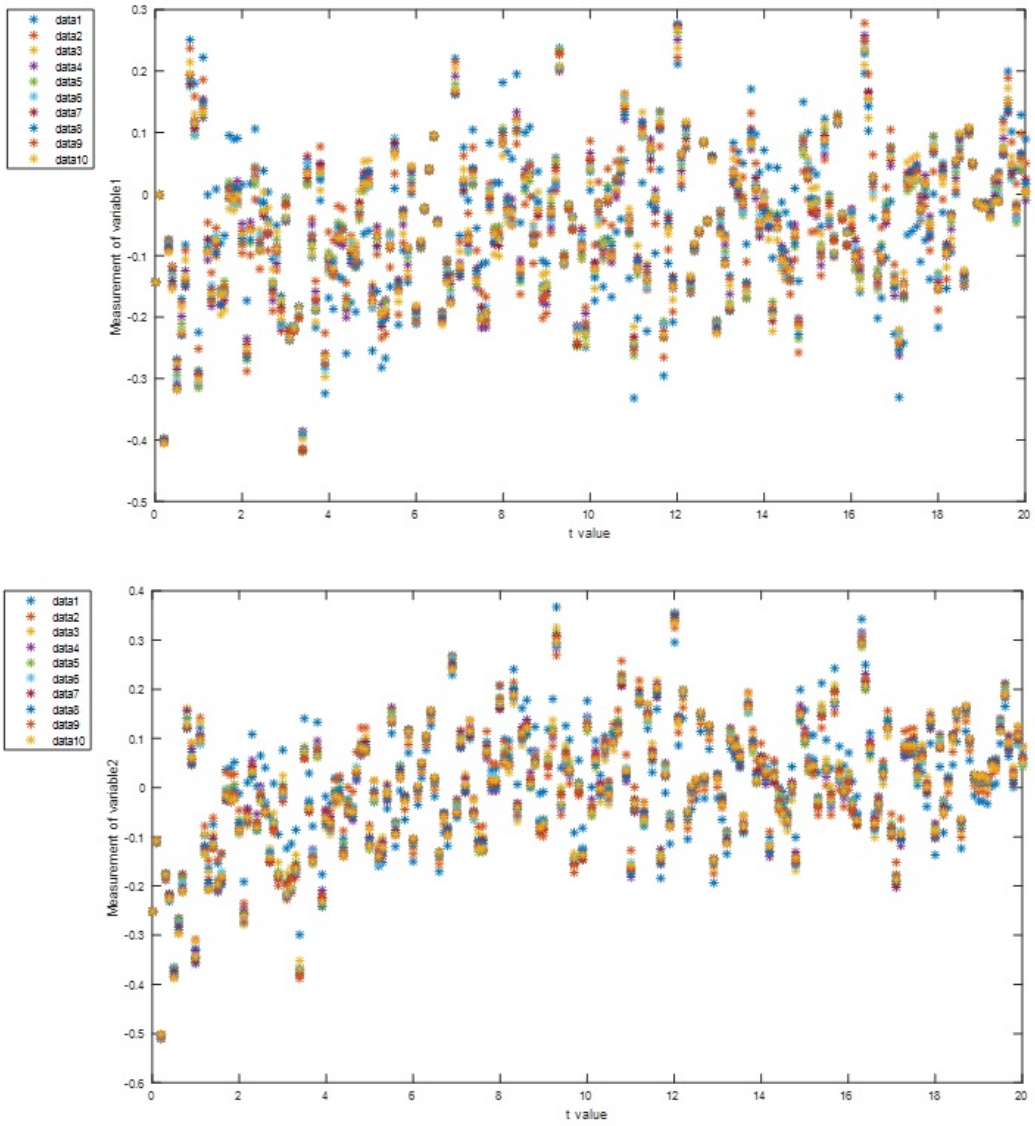

Figure 5: Measured data for Case 2, 10 experiments

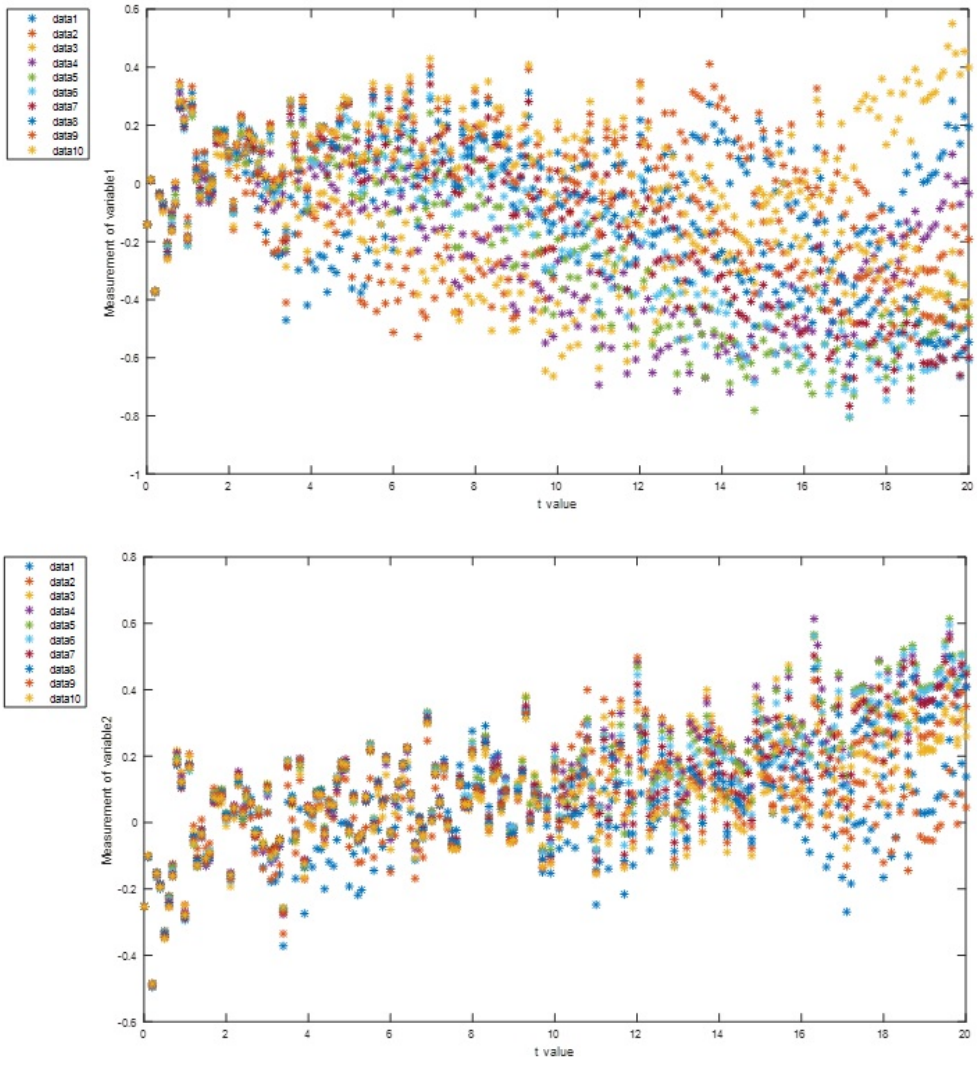

Figure 6: Measured data for Case 3, 10 experiments

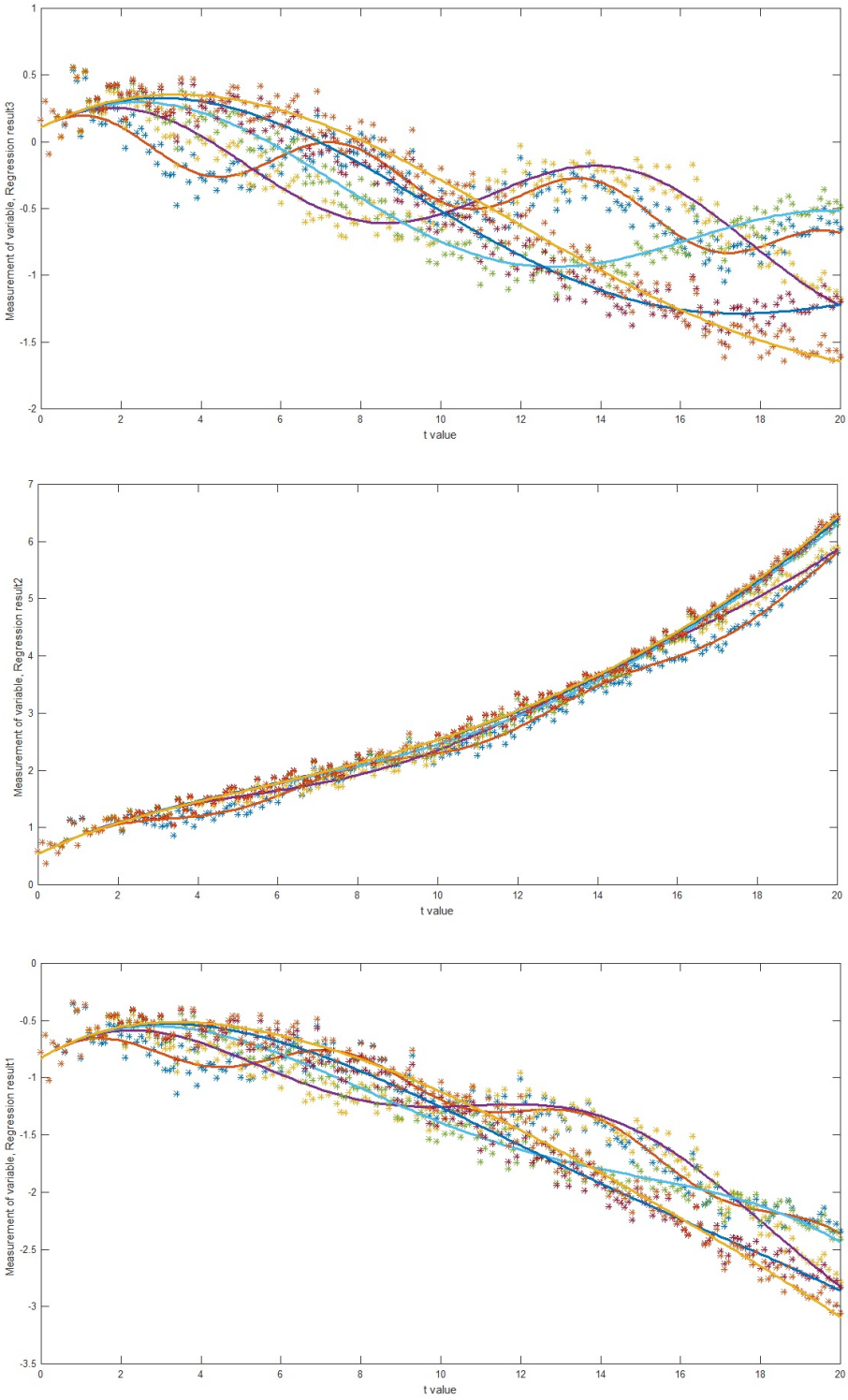

Figure 7: Measured data and Model match for Case 1, 5 experiments

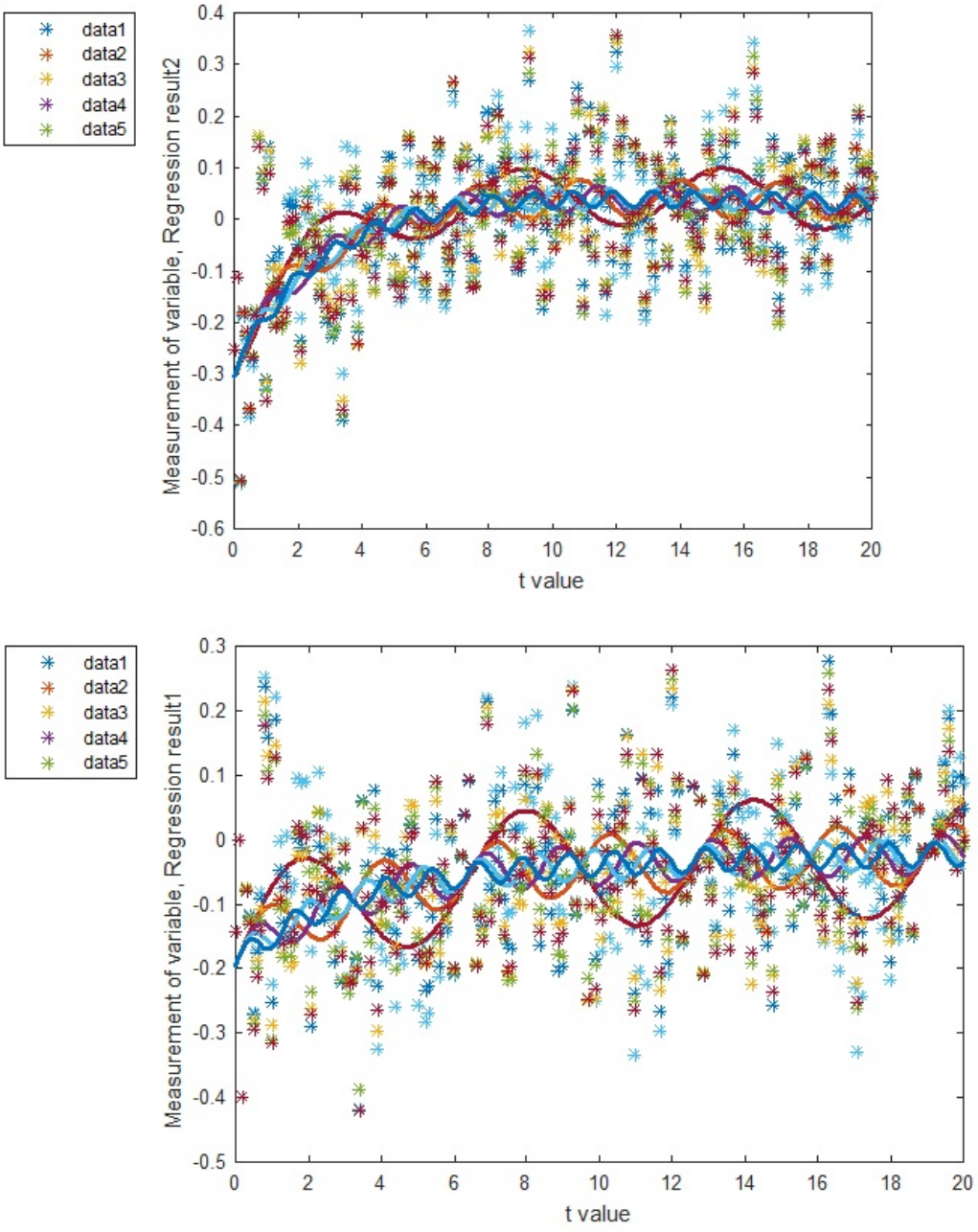

Figure 8: Measured data and Model match for Case 2, 5 experiments

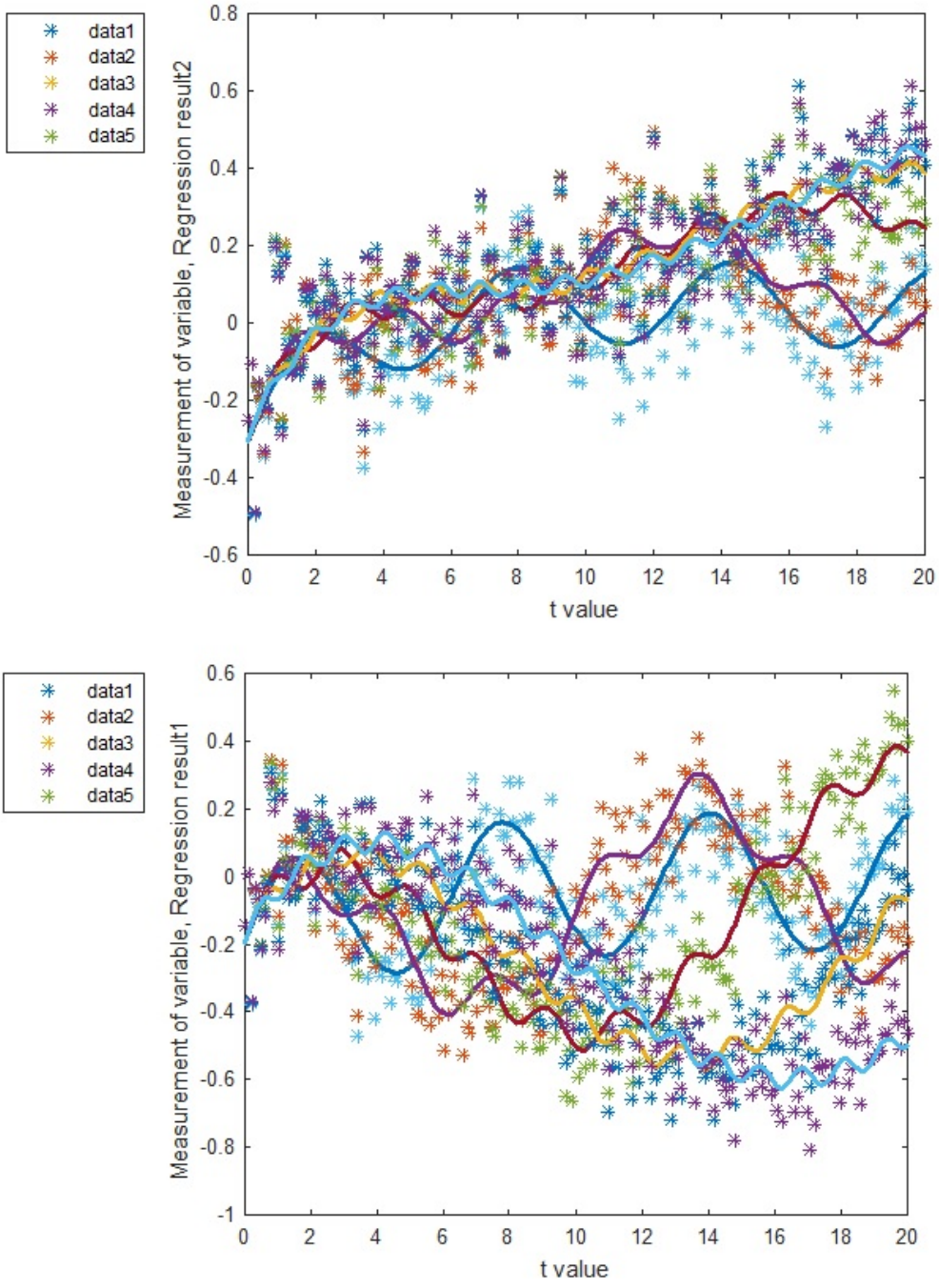

Figure 9: Measured data and Model match for Case 3, 5 experiments