SC42025 Filtering & Identification

MATLAB EXERCISE HW3

Instructions: Fill in the live script with your code and answers. Then export the live script as a pdf (in the 'Live Editor tab', click on the arrow under 'Save' and then 'Export to pdf').

Table of Contents

Consider the following Output Error system:

$$y(k) = \frac{b_1 q^{-1} + b_2 q^{-2} + b_3 q^{-3}}{1 + a_1 q^{-1} + a_2 q^{-2} + a_3 q^{-3} + a_4 q^{-4} + a_5 q^{-5}} u(k) + e(k).$$

```
close all; clear; clc;
addpath('./function_folder')
```

```
Warning: Name is nonexistent or not a directory: C:\Users\linxi\Documents\MATLAB\Homework03
(3)\function_folder\.\function_folder
```

```
load iodata.mat
```

Questions

a) Parameterize the system using the observable canonical form.

Answer: The parameterization of system in observable canonical form is written as

$$x(k+1) = \begin{bmatrix} a_1 & 1 & 0 & 0 & 0 \\ a_2 & 0 & 1 & 0 & 0 \\ a_3 & 0 & 0 & 1 & 0 \\ a_4 & 0 & 0 & 0 & 1 \\ a_5 & 0 & 0 & 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ 0 \\ 0 \end{bmatrix} u(k) \qquad (1)$$

$$y(k) = [1 \quad 0 \quad 0 \quad 0 \quad 0] x(k) + e(k) \qquad (2)$$

b) How many parameters do we have in part (a) ($\theta \in \mathbb{R}^p$)? Represent as $\theta = \begin{pmatrix} \theta_A \\ \theta_B \\ \theta_{x_0} \end{pmatrix}$.

Answer: We have 13 parameters in part (a), including $a_1, a_2, a_3, a_4, a_5, b_1, b_2, b_3$ and $x_{10}, x_{20}, x_{30}, x_{40}, x_{50}$ (x_0 is 5×1 dimensional), stacking all the parameters in θ, we have:

$$\theta = \begin{pmatrix} \theta_A \\ \theta_B \\ \theta_{x_0} \end{pmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & b_1 & b_2 & b_3 & x_0 \end{bmatrix}^T \in R^{13\times 1}, \text{ where } \theta_A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \in R^{5\times 1},\ \theta_B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \in R^{3\times 1} \text{ and}$$

$$\theta_{x_0} = x_0 = \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \\ x_{40} \\ x_{50} \end{bmatrix} \in R^{5\times 1}.$$

c) We are now going to implement a Prediction Error method (`pem.m`) for the Output Error system. We will do this in the following four steps:

I. Derive expressions for $\frac{\partial \overline{A}(\theta)}{\partial \theta_p^{(i)}}, \frac{\partial \overline{B}(\theta)}{\partial \theta_p^{(i)}}, \frac{\partial K(\theta)}{\partial \theta_p^{(i)}}, \frac{\partial C(\theta)}{\partial \theta_p^{(i)}}, \frac{\partial D(\theta)}{\partial \theta_p^{(i)}}, \frac{\partial x_0(\theta)}{\partial \theta_p^{(i)}}$ (for all p).

$$\frac{\partial \overline{A}(\theta)}{\partial \theta_1^{(i)}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \frac{\partial \overline{A}(\theta)}{\partial \theta_2^{(i)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \frac{\partial \overline{A}(\theta)}{\partial \theta_3^{(i)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \frac{\partial \overline{A}(\theta)}{\partial \theta_4^{(i)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\frac{\partial \overline{A}(\theta)}{\partial \theta_5^{(i)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \frac{\partial \overline{A}(\theta)}{\partial \theta_6^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_7^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_8^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_9^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_{10}^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_{11}^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_{12}^{(i)}} = \frac{\partial \overline{A}(\theta)}{\partial \theta_{13}^{(i)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \overline{B}(\theta)}{\partial \theta_6^{(i)}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \frac{\partial \overline{B}(\theta)}{\partial \theta_7^{(i)}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \frac{\partial \overline{B}(\theta)}{\partial \theta_8^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$\frac{\partial \overline{B}(\theta)}{\partial \theta_1^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_2^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_3^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_4^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_5^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_9^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_{10}^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_{11}^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_{12}^{(i)}} = \frac{\partial \overline{B}(\theta)}{\partial \theta_{13}^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$K(\theta) = 0, \ \frac{\partial K(\theta)}{\partial \theta_p^{(i)}} = 0, \text{ for } 1 \le p \le 13$$

$$C(\theta) = [1 \quad 0 \quad 0 \quad 0 \quad 0], \frac{\partial C(\theta)}{\partial \theta_p^{(i)}} = [0 \quad 0 \quad 0 \quad 0 \quad 0],$$

$$D(\theta) = 0, \frac{\partial D(\theta)}{\partial \theta_p^{(i)}} = 0, \text{ for } 1 \leq p \leq 13$$

$$\frac{\partial x_0(\theta)}{\partial \theta_9^{(i)}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \frac{\partial x_0(\theta)}{\partial \theta_{10}^{(i)}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \frac{\partial x_0(\theta)}{\partial \theta_{11}^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \frac{\partial x_0(\theta)}{\partial \theta_{12}^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \frac{\partial x_0(\theta)}{\partial \theta_{13}^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\frac{\partial x_0(\theta)}{\partial \theta_1^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_2^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_3^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_4^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_5^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_6^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_7^{(i)}} = \frac{\partial x_0(\theta)}{\partial \theta_8^{(i)}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

II. Based on the result in part (a), implement a MATLAB function that computes the state-space matrices from the parameter vector θ. (*Look at* `theta2matrices.m` *in the provided template*).

```
[~] = test_theta2matrices;
```

```
Nice job! theta2matrices is correct!
```

III. Write a function `simsystem.m` that simulates a dynamic system given any input vector u and matrices A, B, C, D, as well as initial condition $x(0)$.

```
[~] = test_simsystem;
```

```
Nice job! simsystem is correct!
```

IV.

1. Implement a function `jacobian.m` that calculates the Jacobian vector.

```
[~] = test_jacobian;
```

```
Nice job! jacobian is correct!
```

1. Implement a function `hessian.m` that calculates the approximated Hessian matrix.

```
[~] = test_hessian;
```

```
Nice job! hessian is correct!
```

Note: The convergence loop for the regularized Gauss-Newton algorithm is already implemented in the `pem.m` function. There is no reason to edit this.

d) Choose two different initial guesses for θ. For each guess, train two models such that the first model is obtained from the first 5000 samples and the second one is obtained from the first 10000 samples (from the given iodata.mat). Then, apply all four identified models on **the validation set** and plot the predicted output of all models (only for the validation set) in one figure. Add the numerical values of the VAF and the RMSE to the legend of this figures. (Do not forget to label the models properly!)

```
% split data

ut1 = u(1:5000);            % input for train-set 1
ut2 = u(1:10000);           % input for train-set 2
uv = u(10001:end);          % input for validation-set

yt1 = ymeas(1:5000);        % output for train-set 1
yt2 = ymeas(1:10000);       % output for train-set 2
yv = ymeas(10001:end);      % output for validation-set
```

```
lambda = 1000;
maxiter = 500;

%theta set 1
theta1 = [-1/20 -1/1000 -1/100000 -1/20000000 -1/10000000000 3/2 3/4 1/8 0*ones(1,5)]';
[A01,B01,C01,D01,x0i1] = theta2matrices(theta1);

%5k data
[A11,B11,C11,D11,x0f11,J11,H11] = pem(theta1,A01,B01,C01,D01,x0i1,yt1,ut1,lambda,maxiter);
[y5kt1,~] = simsystem(A11,B11,C11,D11,x0f11,uv);
rms5kt1 = rms(yv-y5kt1)
```

```
rms5kt1 = 0.7844
```

```
%10k data
[A12,B12,C12,D12,x0f12,J12,H12] = pem(theta1,A01,B01,C01,D01,x0i1,yt2,ut2,lambda,maxiter);
[y10kt1,~] = simsystem(A12,B12,C12,D12,x0f12,uv);
rms10kt1 = rms(yv-y10kt1)
```

```
rms10kt1 = 0.7793
```

```
vaf10kt1 = max(0,(1 - rms(yv-y10kt1)/rms(yv))*100)
```

```
vaf10kt1 = 87.7270
```

```
%theta set 2
theta2 = [1/4 -1/200 1/20000 -1/4000000 1/2000000000 3/2 3/4 1/8 1*ones(1,5)]';
[A02,B02,C02,D02,x0i2] = theta2matrices(theta2);

%5k data
```

```
[A21,B21,C21,D21,x0f21,J21,H21] = pem(theta2,A02,B02,C02,D02,x0i2,yt1,ut1,lambda,maxiter);
[y5kt2,~] = simsystem(A21,B21,C21,D21,x0f21,uv);
rms5kt2 = rms(yv-y5kt2)
```

```
rms5kt2 = 0.6687
```

```
vaf5kt2 = max(0,(1 - rms(yv-y5kt2)/rms(yv))*100)
```

```
vaf5kt2 = 89.4688
```

```
%10k data
[A22,B22,C22,D22,x0f22,J22,H22] = pem(theta2,A02,B02,C02,D02,x0i2,yt2,ut2,lambda,maxiter);
[y10kt2,~] = simsystem(A22,B22,C22,D22,x0f22,uv);
rms10kt2 = rms(yv-y10kt2)
```

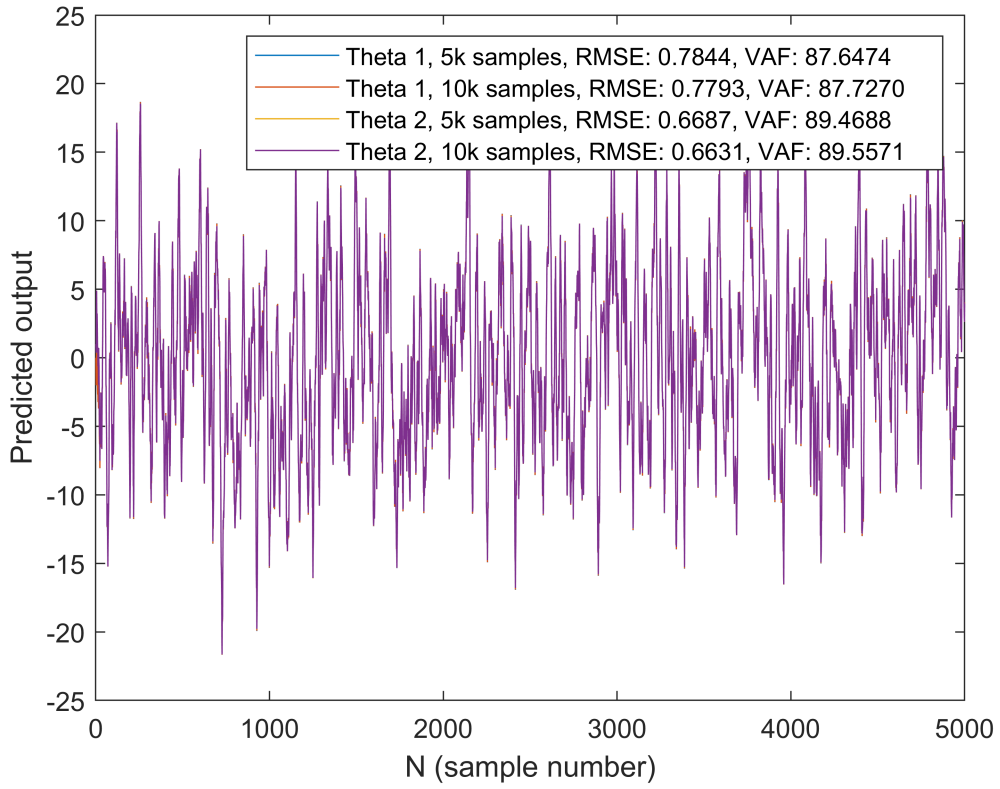
```
rms10kt2 = 0.6631
```

```
vaf10kt2 = max(0,(1 - rms(yv-y10kt2)/rms(yv))*100)
```

```
vaf10kt2 = 89.5571
```

```
close all
plot(y5kt1)
hold on
plot(y10kt1)
plot(y5kt2)
plot(y10kt2)

xlabel('N (sample number)')
ylabel('Predicted output')
legend('Theta 1, 5k samples, RMSE: 0.7844, VAF: 87.6474',...
    'Theta 1, 10k samples, RMSE: 0.7793, VAF: 87.7270',...
    'Theta 2, 5k samples, RMSE: 0.6687, VAF: 89.4688',...
    'Theta 2, 10k samples, RMSE: 0.6631, VAF: 89.5571')
```



e) According to the result in (d), report the system parameters a_1, a_2, a_3, a_4, a_5, b_1, b_2, b_3 for the best identified model. Explain how you chose the best model.

Answer: The system parameters a_1, a_2, a_3, a_4, a_5, b_1, b_2, b_3 for the best identified model are 0.632, 0.348, 0.155, -0.035, -0.194, 1.427, 0.799 and 0.284 respectively. This model is obtained by using the second initial guess θ_2 and fisrt 10000 samples, because it has the least RMSE and the largest VAF compared with the other three models. RMSE represents the standard deviation of the errors which the model make compared the validation data set. The smaller the RMSE of the model means the more concentrated around the real value it can eastimate overally. The VAF of two models that are the same is 100%. If they differ, the VAF will be lower. By comparing the real output with the estimated output of the model, the VAF can be used to verify the correctness of a model. Thus we choose the model with the least RMSE and the largest VAF to be the best identified model.

Functions

Implement your functions in the templates below.

You may implement additional functions for your convenience.

```
function [Abar,Bbar,C,D,x0] = theta2matrices(theta)
```

```
% Function INPUT
% theta     Paramter vector (vector of size 'number of parameters' x 1)

% Function OUTPUT
% Abar      System matrix A (matrix of size n x n)
% Bbar      System matrix B (matrix of size n x m)
% C         System matrix C (matrix of size l x n)
% D         System matrix D (matrix of size l x m)
% x0        Initial state (vector of size n x one)

% YOUR CODE HERE

Abar = [theta(1), 1, 0, 0, 0;
    theta(2), 0, 1, 0, 0;
    theta(3), 0, 0, 1, 0;
    theta(4), 0, 0, 0, 1;
    theta(5), 0, 0, 0, 0];

Bbar = [theta(6); theta(7); theta(8); 0; 0];
C = [1, 0, 0, 0, 0];
D = 0;
x0 = [theta(9); theta(10); theta(11); theta(12); theta(13)];

end
```

```
function [y, x] = simsystem(A, B, C, D, x0, u)
% Instructions:
% Simulating a linear dynamic system given input u, matrices A,B,C,D ,and
% initial condition x(0)
%
%
% Function INPUT
% A system matrix (matrix of size n x n)
% B system matrix (matrix of size n x m)
% C system matrix (matrix of size l x n)
% D system matrix (matrix of size l x m)
% x0 initial state (vector of size n x one)
% u system input (matrix of size N x m)
%
% Function OUTPUT
% x state of system (vector of size N x n)
% y system output (matrix of size N x l)
 n = size(A, 1);
 m = size(B, 2);
 l = size(C, 1);
 N = size(u, 1);
x = zeros(N,n);
```

```
y = zeros(N,l);

x(1,:) = x0';
y(1,:) = C*x(1,:)'+D*u(1,:)';

for i = 2:1:N
    x(i,:) = A*x(i-1,:)'+B*u(i-1,:)';
    y(i,:) = C*x(i,:)'+D*u(i,:)';

end


end
```

```
function J = jacobian(psi,E)
% Instructions:
% This function calculates the Jacobian vector in the Gauss-Newton
% algorithm
%
%  p = size(theta);
%  N = size(y, 1);
%  l = size(C, 1);
%
% Function INPUT
% psi derivative of estimation error wrt theta (matrix of size l*N x p)
% E estimation error vector (vector of size l*N x one)
%
% Function OUTPUT
% J Jacobian vector (vector of size p x one)

l = 1;
N=size(psi,1)/l;
J=2/N*psi'*E;


end
```

```
function H = hessian(psi)
% Instructions:
% This function calculates the approximated Hessian matrix in the Gauss-Newton
% algorithm
%
%  p = size(theta);
```

```
%  N = size(y, 1);
%  l = size(C, 1);
%
% Function INPUT
% psi derivative of estimation error wrt theta (matrix of size l*N x p)
%
% Function OUTPUT
% H Hessian matrix (matrix of size p x p)

l = 1;
N=size(psi,1)/l;
H = 2/N*(psi'*psi);

end
```

```
function [Abar,Bbar,C,D,x0, J, H] = pem(theta,A0,B0,C0,D0,x00,y,u,lambda,maxiter)
% Instructions:
% Implement your Prediction Error Method for the Output Error system here.
% Use the following function inputs and outputs.
%
% Function INPUT
% theta  Paramter vector (size: depends on your mapping choice)
% A0 Initial guess for system matrix A (matrix of size n x n)
% B0 Initial guess for system matrix B (matrix of size n x m)
% C0 Initial guess for system matrix C (matrix of size l x n)
% D0 Initial guess for system matrix D (matrix of size l x m)
% x00 Initial guess for initial state (vector of size n x one)
% u System input (matrix of size N x m)
% y System output (matrix of size N x l)
% lambda regularization parameter (scalar)
% maxiter Maximum number of iterations (scalar)
%
%
% Function OUTPUT
% Abar Estimate of system matrix A (matrix of size n x n)
% Bbar Estimate of system matrix B (matrix of size n x m)
% C Estimate of system matrix C (matrix of size l x n)
% D Estimate of system matrix D (matrix of size l x m)
% x0 Estimate of initial state (vector of size n x one)


% loop counters
p = length(theta);
N = length(u);

%initialization
A = A0;
```

```
B = B0;
C = C0;
D = D0;
x0 = x00;

% Gauss-Newton

% initialize
converged = false;
maxiter_reached = false;
iter = 1;

while ~converged && ~maxiter_reached

    %calculation of error vector
    [y_hat, x_hat] = simsystem(A,B,C,D,x0,u);
    E = y - y_hat;

    %psi calculation
    psi = zeros(N,p); %preallocation

    for j = 1:p

        [dadtp,dbdtp,dx0dtp] = pthderivative(j);

        [dydtp, ~] = simsystem(A,[dadtp dbdtp],C,zeros(1,6),dx0dtp,[x_hat u]);

        psi(:,j) = -1*dydtp;

    end

    J = jacobian(psi,E);
    H = hessian(psi);

    theta_new = theta - (H + lambda*eye(size(H,1))) \ J;
    [A,B,C,D,x0]=theta2matrices(theta_new);

    % Check convergence
    if norm(theta_new - theta) < 1e-3
        converged = true;
    elseif iter == maxiter
        maxiter_reached = true;
        warning('Maximum iterations reached');
    end

    theta = theta_new;
    iter = iter + 1;

end
```

```
% YOUR CODE HERE

Abar = A;
Bbar = B;

end

function [dadtp, dbdtp, dx0dtp] = pthderivative(p)

    switch p

        case 1
            dadtp = zeros(5,5); dadtp(1,1) = 1;
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1);
        case 2
            dadtp = zeros(5,5); dadtp(2,1) = 1;
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1);
        case 3
            dadtp = zeros(5,5); dadtp(3,1) = 1;
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1);
        case 4
            dadtp = zeros(5,5); dadtp(4,1) = 1;
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1);
        case 5
            dadtp = zeros(5,5); dadtp(5,1) = 1;
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1);
        case 6
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1); dbdtp(1,1) = 1;
            dx0dtp = zeros(5,1);
        case 7
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1); dbdtp(2,1) = 1;
            dx0dtp = zeros(5,1);
        case 8
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1); dbdtp(3,1) = 1;
            dx0dtp = zeros(5,1);
        case 9
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1); dx0dtp(1,1) = 1;
        case 10
            dadtp = zeros(5,5);
```

```
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1); dx0dtp(2,1) = 1;
        case 11
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1); dx0dtp(3,1) = 1;
        case 12
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1); dx0dtp(4,1) = 1;
        case 13
            dadtp = zeros(5,5);
            dbdtp = zeros(5,1);
            dx0dtp = zeros(5,1); dx0dtp(5,1) = 1;
        otherwise
            warning('unexpected behaviour')
    end
end
```