SANDIA REPORT

SAND2017-4401
Unlimited Release
Printed April 2017

State estimation for wave energy converters

Giorgio Bacelli and Ryan G. Coe

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-mission laboratory
managed and operated by National Technology and Engineering Solutions of Sandia, LLC.,
a wholly owned subsidiary of Honeywell International, Inc.,
for the U.S. Department of Energy's National Nuclear Security Administration
under contract DE-NA0003525.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology and Engineering Solutions of Sandia, LLC..

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2017-4401
Unlimited Release
Printed April 2017

State estimation for wave energy converters

Giorgio Bacelli
Water Power Technologies Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
gbacell@sandia.gov

Ryan G. Coe
Water Power Technologies Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-9999
rcoe@sandia.gov

Contents

Appendix

Figures

1 Introduction

This report gives a brief discussion and examples on the topic of state estimation for wave energy converters (WECs). These methods are intended for use to enable real-time closed loop control of WECs. The algorithm for the optimal estimation of unknown inputs and state requires the system to be expressed using a discrete-time state-space model as [1]

$$x_{k+1} = Ax_k + Bu_k + Gd_k + w_k \tag{1a}$$

$$y_k = Cx_k + Du_k + Hd_k + v_k, \tag{1b}$$

where d_k is the unknown input at time step k (i.e. the excitation force $d_k = F_{e_k}$ or the excitation pressure $d_k = P_{e_k}$), u_k is the known input (i.e. the actuator's force $u_k = F_{a_k}$), y_k is the measurements vector (i.e. acceleration and pressure $y = [\ddot{z}, P]^T$), w_k and v_k are the process and measurements noise, respectively. Two examples are discussed here: using position and acceleration measurements (Section 2) and using pressure and acceleration measurements (Section 3).

2 Using position

This Section provides an example for using position and acceleration to predict the state of a WEC. Sample code for implementing this example is provided in Appendix A. The block diagram of the buoy's dynamic model is depicted in Fig. 1. The "intrinsic" model of the buoy G_i in continuous-time state space form is

$$\dot{x}_i = A_i x_i + B_i (F_e + F_a) \tag{2a}$$
$$y_i = C_i x_i + D_i (F_e + F_a) \tag{2b}$$

where the state vector is

$$x = \begin{bmatrix} z \\ \dot{z} \\ x_r \end{bmatrix}, \tag{3}$$

and

$$A_i = \begin{bmatrix} -\frac{B_f}{m+m_\infty} & -\frac{K}{m+m_\infty} & -\frac{C_r}{m+m_\infty} \\ 1 & 0 & \mathbf{0}_{1\times n_r} \\ B_r & \mathbf{0}_{n_r\times 1} & A_r \end{bmatrix} \qquad B_i = \begin{bmatrix} \frac{1}{m+m_\infty} \\ 0 \\ \mathbf{0}_{n_r\times 1} \end{bmatrix} \tag{4}$$

$$C_i = \begin{bmatrix} 0 & 1 & \mathbf{0}_{1\times n_r} \\ -\frac{B_f}{m+m_\infty} & -\frac{K}{m+m_\infty} & -\frac{C_r}{m+m_\infty} \end{bmatrix} \qquad D_i = \begin{bmatrix} 0 \\ \frac{1}{m+m_\infty} \end{bmatrix} \tag{5}$$

and where the matrices $A_r \in \mathbb{R}^{n_r\times n_r}$, $B_r \in \mathbb{R}^{n_r\times 1}$ and $C_r \in \mathbb{R}^{1\times n_r}$ describe the radiation force F_r dynamics as

$$\dot{x}_r = A_r x_r + B_r \dot{z} \tag{6a}$$
$$F_r = C_r x_r. \tag{6b}$$

The mass of the buoy is denoted by m, the hydrostatic restoring coefficient by K, the friction coefficient by B_f and m_∞ is the asymptotic value of the added mass for $\omega \to \infty$. Two steps are now required to formulate system in (2) as required in (1)

1. Convert to discrete time
2. Derive matrices A, B, C, D, G and H

If the matrix A_i is not singular, then step 1 can be carried out by using the

$$A = e^{A_i T_c} \tag{7}$$
$$B = A_i^{-1} (A - I) B_i \tag{8}$$

where T_c is the sampling time. The matrices C, D, G and H are:

$$C = C_i \qquad D = D_i \tag{9}$$
$$G = B \qquad H = D. \tag{10}$$

The time-varying version of the algorithm given in 1 whereas the steady-state version (much faster computation) is given in 2.

Algorithm 1 Time-varying Unknown Input and State Estimator.

▷ Initialize:
$\hat{x}_{0|0} = \mathbb{E}[x_0]$
$\hat{d}_0 = H^{\dagger}\left(y_0 - C\hat{x}_{0|0} - Du_0\right)$
$P^x_{0|0} = \mathscr{P}^x_0$
$P^d_0 = \mathscr{P}^d_0$
$P^{xd}_0 = \mathscr{P}^{xd}_0$
$Q = \mathbb{E}[w\,w^T]$
$R = \mathbb{E}[v\,v^T]$
▷ Estimation loop for N time steps (Time step = T_c)
for $k = 1$ to N **do**
▷ One-Step prediction
  $\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1} + G\,\hat{d}_{k-1}$
  $P^x_{k|k-1} = A\,P^x_{k-1|k-1}A^T + G\,P^{xd^T}_{k-1}A^T + A\,P^{xd}_{k-1}G^T + G\,P^d_{k-1}G^T + Q$
  $\tilde{R}_k = C\,P^x_{k|k-1}C^T$
▷ Measurements update
  $K_k = P^x_{k|k-1}C^T\,\tilde{R}_k^{-1}$
  $L_k = K_k\left(I - H\left(H^T\,\tilde{R}_l^{-1}H\right)^{-1}H^T\,\tilde{R}_k^{-1}\right)$
  $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k\left(y_k - C\,\hat{x}_{k|k-1} - D\,u_k\right)$
  $P^x_{k|k} = (I - L_k C)\,P^x_{k|k-1}\,(I - L_k C)^T + L_k R L_k^T$
▷ Estimation of unknown input
  $\tilde{R}^*_k = (I - CL_k)\,\tilde{R}_k\,(I - CL_k)^T$
  $P^d_k = \left(H^T\tilde{R}^{*-1}_k H\right)^{-1}$
  $M_k = P^d_k H^T\tilde{R}^{*-1}$
  $\hat{d}_k = M_k\left(y_k - C\hat{x}_{k|k} - D\,u_k\right)$
  $P^{xd}_k = -P^x_{k|k}C^T M_k^T + L_k R M_k^T$
end for

Algorithm 2 Steady-State Unknown Input and State Estimator.

▷ Initialize:
$\hat{x}_{0|0} = \mathbb{E}[x_0]$
$\hat{d}_0 = H^{\dagger}\left(y_0 - C\hat{x}_{0|0} - Du_0\right)$
$L_\infty = \lim_{k\to\infty} L_k$
$M_\infty = \lim_{k\to\infty} M_k$
▷ Estimation loop for N time steps (Time step = T_c)
for $k = 1$ to N **do**
  $\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1} + G\,\hat{d}_{k-1}$ ▷ One-Step prediction
  $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_\infty\left(y_k - C\,\hat{x}_{k|k-1} - D\,u_k\right)$ ▷ State estimation
  $\hat{d}_k = M_\infty\left(y_k - C\hat{x}_{k|k} - D\,u_k\right)$ ▷ Unknown input estimation
end for

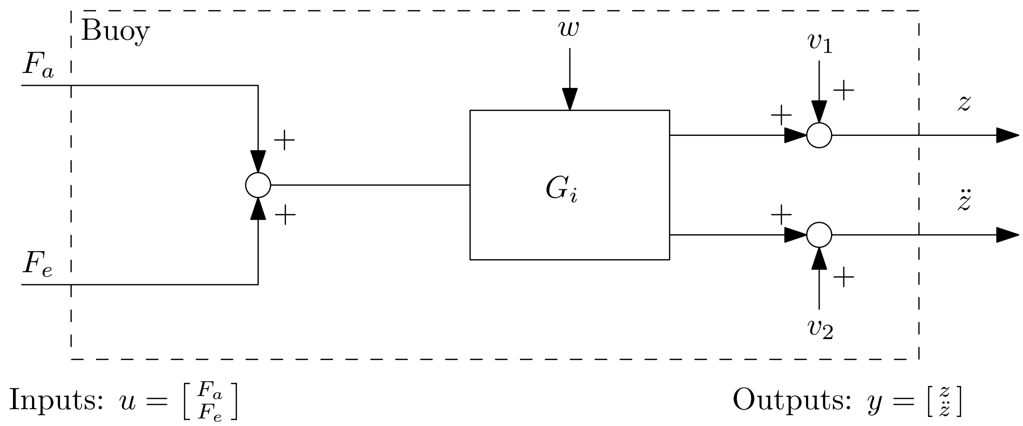

Figure 1. Block diagram of the buoy's dynamic model when position and acceleration are measured

3 Using pressure

This Section provides an example for using pressure and acceleration to predict the state of a WEC. Sample code for implementing this example is provided in Appendix B. The block diagram of the buoy's dynamic model is depicted in Fig. 1. The "intrinsic" model of the buoy G_i in continuous-time state space form is

$$\dot{x}_i = A_i x_i + B_i \left(F_e + F_a\right) \tag{11a}$$
$$y_i = C_i x_i + D_i \left(F_e + F_a\right) \tag{11b}$$

where the output vector y_i is

$$y_i = \begin{bmatrix} \ddot{z} \\ P_r \end{bmatrix}, \tag{12}$$

and the matrices composing the state space model in (11) have been identified form experimental data.

The state-space model of the excitation pressure in continuous-time (G_e) is:

$$\dot{x}_e = A_e x_e + B_e P_e \tag{13a}$$
$$F_e = C_e x_e + D_e P_e. \tag{13b}$$

According to the diagram in Fig. 2 the models in (11) and (13) can be combined to form the state-state space model

$$\dot{x} = A_c x + B_c \begin{bmatrix} F_a \\ P_e \end{bmatrix} \tag{14a}$$
$$y = C_c x + D_c \begin{bmatrix} F_a \\ P_e \end{bmatrix} \tag{14b}$$

where the state vector is

$$x = \begin{bmatrix} x_i \\ x_e \end{bmatrix} \tag{15}$$

and where the output vector is

$$y = \begin{bmatrix} \ddot{z} \\ P \end{bmatrix} = \begin{bmatrix} \ddot{z} \\ P_r + P_e \end{bmatrix} = \begin{bmatrix} \ddot{z} \\ P_r \end{bmatrix} + \begin{bmatrix} 0 \\ P_e \end{bmatrix} = y_i + \begin{bmatrix} 0 \\ P_e \end{bmatrix}. \tag{16}$$

The system matrices are

$$A_c = \begin{bmatrix} A_i & B_i C_e \\ \mathbf{0} & A_e \end{bmatrix} \qquad B_c = \begin{bmatrix} B_i & B_u D_e \\ \mathbf{0} & B_e \end{bmatrix} \tag{17}$$

$$C_c = \begin{bmatrix} C_i & D_i C_e \end{bmatrix} \qquad D_c = \begin{bmatrix} D_i & D_i D_e + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} \tag{18}$$

Two steps are now required to formulate system in (14) as required in (1):

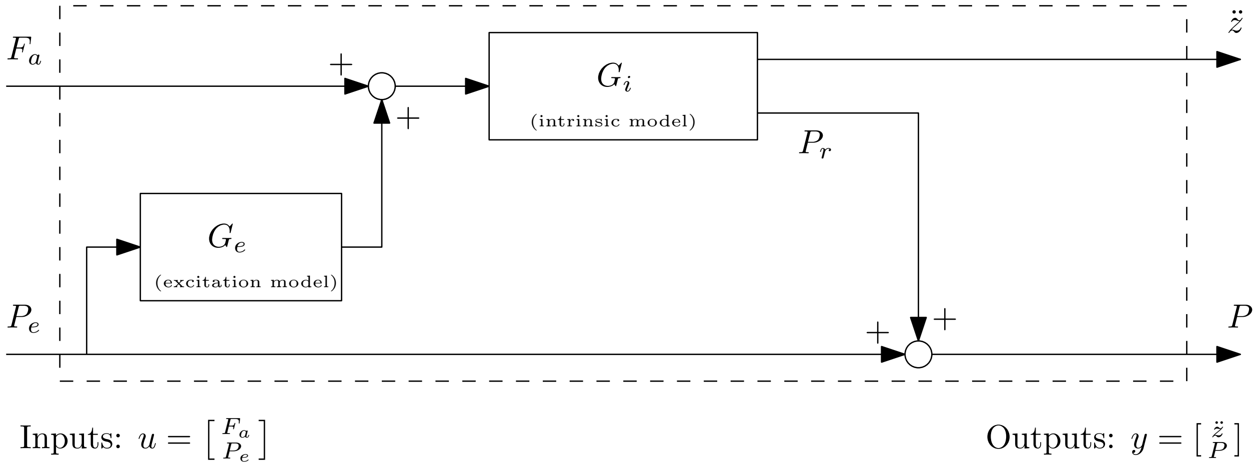

Figure 2. Block diagram of the buoy's dynamic model when pressure and acceleration are measured.

1. Convert to discrete time
2. Derive matrices A, B, C, D, G and H

If the matrix A_c is not singular, then step 1 can be carried out by using the

$$A = e^{A_c T_c} \tag{19}$$

$$\bar{B} = A_c^{-1}(A - I)B_c \tag{20}$$

where T_c is the sampling time. $\bar{B}$ and D_c are $2 \times n$ matrices: B is the first column of $\bar{B}$, G is the second column of $\bar{B}$, D is the first column of D_c and H is the second column of D_c, that is:

$$\bar{B} = [B \quad G] \tag{21}$$

$$C = C_c \qquad\qquad D_c = [D \quad H] \tag{22}$$

$$\tag{23}$$

The time-varying version of the algorithm given in 3 whereas the steady-state version (much faster computation) is given in 4.

Algorithm 3 Time-varying Unknown Input and State Estimator.

▷ Initialize:

$\hat{x}_{0|0} = \mathbb{E}[x_0]$

$\hat{d}_0 = H^{\dagger}\left(y_0 - C\hat{x}_{0|0} - Du_0\right)$

$P^x_{0|0} = \mathscr{P}^x_0$

$P^d_0 = \mathscr{P}^d_0$

$P^{xd}_0 = \mathscr{P}^{xd}_0$

$Q = \mathbb{E}[w\,w^T]$

$R = \mathbb{E}[v\,v^T]$

▷ Estimation loop for N time steps (Time step = T_c)

for $k = 1$ to N **do**

▷ One-Step prediction

$\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1} + G\,\hat{d}_{k-1}$

$P^x_{k|k-1} = A\,P^x_{k-1|k-1}\,A^T + G\,P^{xd^T}_{k-1}\,A^T + A\,P^{xd}_{k-1}\,G^T + G\,P^d_{k-1}\,G^T + Q$

$\tilde{R}_k = C\,P^x_{k|k-1}\,C^T$

▷ Measurements update

$K_k = P^x_{k|k-1}\,C^T\,\tilde{R}_k^{-1}$

$L_k = K_k\left(I - H\left(H^T\,\tilde{R}_l^{-1}H\right)^{-1}H^T\,\tilde{R}_k^{-1}\right)$

$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k\left(y_k - C\,\hat{x}_{k|k-1} - D\,u_k\right)$

$P^x_{k|k} = (I - L_k\,C)\,P^x_{k|k-1}\,(I - L_k\,C)^T + L_k\,R\,L_k^T$

▷ Estimation of unknown input

$\tilde{R}^*_k = (I - CL_k)\,\tilde{R}_k\,(I - CL_k)^T$

$P^d_k = \left(H^T\tilde{R}^{*-1}_k H\right)^{-1}$

$M_k = P^d_k H^T \tilde{R}^{*-1}$

$\hat{d}_k = M_k\left(y_k - C\hat{x}_{k|k} - D\,u_k\right)$

$P^{xd}_k = -P^x_{k|k} C^T M_k^T + L_k R M_k^T$

end for

Algorithm 4 Steady-State Unknown Input and State Estimator.

▷ Initialize:

$\hat{x}_{0|0} = \mathbb{E}[x_0]$

$\hat{d}_0 = H^{\dagger}\left(y_0 - C\hat{x}_{0|0} - Du_0\right)$

$L_\infty = \lim_{k\to\infty} L_k$

$M_\infty = \lim_{k\to\infty} M_k$

▷ Estimation loop for N time steps (Time step = T_c)

for $k = 1$ to N **do**

$\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1} + G\,\hat{d}_{k-1}$ ▷ One-Step prediction

$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_\infty\left(y_k - C\,\hat{x}_{k|k-1} - D\,u_k\right)$ ▷ State estimation

$\hat{d}_k = M_\infty\left(y_k - C\hat{x}_{k|k} - D\,u_k\right)$ ▷ Unknown input estimation

end for

References

[1] Sze Zheng Yong, Minghui Zhu, and Emilio Frazzoli. A unified filter for simultaneous input and state estimation of linear discrete-time stochastic systems. *Automatica*, 63:321 – 329, 2016.

A Sample code: position and acceleration measurements

This section contain sample MATLAB code for implementing a position/acceleration state estimator for a WEC.

```
% this scripts is to test the Unified Linear Input and State Estimator
% (ULISE algorithm) described in
%
% S. Z. Yong, M. Zhu, and E. Frazzoli, A unified filter for simultaneous
% input and state estimation of linear discrete-time stochastic systems,
% Automatica, vol. 63, pp. 321 329 , 2016.
%
% Both Time-Varying and Steady-State versions are implemented.
% Position and acceleration measurements are used to estimate state and
% excitation force
%
% Sandia National Laboratories is a multi-mission laboratory managed and
% operated by National Technology and Engineering Solutions of Sandia,
% LLC., a wholly owned subsidiary of Honeywell International, Inc., for the
% U.S. Department of Energy?s National Nuclear Security Administration
% under contract DE-NA0003525.
%
% G. Bacelli, R. Coe
% Sandia National Laboratories
% 2017


clc
clear

% load identified parametric WEC model model
WEC = load('WEC_param_model_1DOF.mat');

Tc = 1e-3; % sampling time
N = 5e4; % number of simulation steps

% load excitation force and interploate
Fe = load('Exc_time_series.mat');
t = (1:N)*Tc;
d = interp1(Fe.t_trim, Fe.Fexc_td, t, 'pchip')*1e-3; %(excitation force in kN)

u = 0.5*sin(2*pi*0.75*t); % control input (PTO force in kN). Open loop, no control implemented

mass = 858.3987;
Ainf_hat = 822.3799;
K = 2.3981e+04;

Ar = WEC.rad_sys.a;
Br = WEC.rad_sys.b;
Cr = WEC.rad_sys.c;

Bf = WEC.B_eq_mat(9);

Ac = [-Bf/(mass + Ainf_hat)    -K/(mass + Ainf_hat)        -Cr/(mass + Ainf_hat);
       1                            0                       zeros(1,length(Ar));
       Br                       zeros(length(Ar),1)             Ar              ];

% input matrix (1e3 factor is used to acccount for the forces that expressed are in kN)
Bc = 1e3*[1/(mass + Ainf_hat);
          0                  ;
          zeros(length(Ar),1)];

% measurements: position and acceleration
```

```
C = [0 1 zeros(1,length(Ar));
       Ac(1,:)                ];

D = [0; Bc(1)];

n = size(C,2);
p = size(C,1);

% convert continuous time model to discrete time
A = expm(Ac*Tc);
B = Ac\(A - eye(n))*Bc;

G = B;
H = D;

% process noise
w = .00001*(rand(N,4)-0.5);
% measurements noise
v = (rand(N,p)-0.5) * diag([.0001,0.05]);

Q = cov(w);
R = cov(v);

%% Time varying filter

% initialize variables
P_x_k_k = 0.001*eye(n);
P_xd_k = 0.001*ones(n,1);
P_d_k = 0.001;

x = zeros(n,1);
x_k_k = x;
d_k = 0;

In = eye(n);
Ip = eye(p);

% preallocation
d_k_vec = zeros(N,1);
x_k_vec = zeros(n,N);
x_vec = zeros(n,N);
y_vec = zeros(p,N);


tic

for ii = 1:N

    x = A*x + B*u(ii) + G*d(ii) + w(ii,:)';
    y = C*x + D*u(ii) + H*d(ii) + v(ii,:)';

    x_vec(:,ii) = x;
    y_vec(:,ii) = y;

    x_k_k1 = A*x_k_k + B*u(ii) + G*d_k;
    P_x_k_k1 = A*P_x_k_k*A' + G*P_xd_k'*A' + A*P_xd_k*G' + G*P_d_k*G' + Q;
    R_t_k = C*P_x_k_k1*C' + R;
    Kk = P_x_k_k1*C'/R_t_k;
    Lk = Kk*(Ip-((H/((H'/R_t_k)*H))*H'/R_t_k ));
    x_k_k = x_k_k1 + Lk*(y - C*x_k_k1 - D*u(ii) );
    P_x_k_k = (In-Lk*C)*P_x_k_k1*(In-Lk*C)' + Lk*R*Lk';
    R_ts_k = (Ip-C*Lk)*R_t_k*(Ip-C*Lk)';
    P_d_k = inv((H'/R_ts_k)*H);
    Mk = ((H'/R_ts_k)*H)\(H'/R_ts_k);
    d_k = Mk*(y - C*x_k_k - D*u(ii));
    P_xd_k = -P_x_k_k*C'*Mk' + Lk*R*Mk';

    d_k_vec(ii) = d_k;
```

```
    x_k_vec(:,ii) = x_k_k;

end

t_tv = toc;
disp('done TV')

%% steady state filter

L_inf = Lk;
M_inf = Mk;

% preallocation
d_k_vec_inf = zeros(N,1);
x_vec_inf = zeros(n,N);
y_vec_inf = zeros(p,N);
x_vec_est_inf = x_vec_inf;

% initialization
x = zeros(n,1);
x_k_k_inf = zeros(n,1);
d_k1_inf = 0;

tic
for ii = 1:N

    x = A*x + B*u(ii) + G*d(ii) + w(ii,:)';
    y = C*x + D*u(ii) + H*d(ii) + v(ii,:)';

    x_vec_inf(:,ii) = x;
    y_vec_inf(:,ii) = y;

    x_k_k1_inf = A*x_k_k_inf + B*u(ii) + G*d_k1_inf;
    x_k_k_inf = x_k_k1_inf + L_inf*(y - C*x_k_k1_inf - D*u(ii) );
    d_k1_inf = M_inf*(y - C*x_k_k_inf - D*u(ii));

    d_k_vec_inf(ii) = d_k1_inf;
    x_vec_est_inf(:,ii) = x_k_k_inf;
end

t_ss = toc;
disp('done SS')


%% plotting

disp(['Time to compute Time-Varying filter: ' num2str(t_tv) 's'])
disp(['Time to compute Steady-State filter: ' num2str(t_ss) 's'])

figure(1)
plot(t, d_k_vec', t, d)
xlabel('time (s)')
ylabel('(kN)')
grid on
title('Time-Varying filter')
legend({'${\hat{F}e}_{\infty}$', '$Fe$'}, 'Interpreter', 'latex')

figure(2)
plot(t, d_k_vec_inf, t, d)
xlabel('time (s)')
ylabel('(kN)')
grid on
title('Steady-State filter')
legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')

figure(3)
plot(t, d_k_vec - d_k_vec_inf)
grid on
```

```
xlabel('Time (s)')
title('Difference between Time-Varying and Steady-State filters')
legend('e_d')

figure(4)
subplot 211
plot(t, y_vec(1,:)')
xlabel('Time (s)')
ylabel('(m)')
title('Measured (noisy) Outputs')
legend({'$z$'}, 'Interpreter', 'latex')
grid on
subplot 212
plot(t, y_vec(2,:)')
xlabel('Time (s)')
ylabel('(m/s^2)')
legend({'$\ddot{z}$'}, 'Interpreter', 'latex')
grid on


figure(5)
subplot 211
plot(t, x_vec(1,:)', t, x_k_vec(1,:)', t, x_vec_est_inf(1,:)')
grid on
xlabel('time (s)')
ylabel('(m/s)')
legend({'$v$', '$\hat{v}$', '$\hat{v}_{\infty}$'}, 'Interpreter', 'latex')
title('Estimated states')
grid on

subplot 212
plot(t, x_vec(2,:)', t, x_k_vec(2,:)', t, x_vec_est_inf(2,:)')
grid on
xlabel('time (s)')
ylabel('(m)')
legend({'$z$', '$\hat{z}$', '$\hat{z}_{\infty}$'}, 'Interpreter', 'latex')
grid on
```

State_and_unknown_input_estimator_position_acceleration.m

B Sample code: pressure and acceleration measurements

This section contain sample MATLAB code for implementing a pressure/acceleration state estimator for a WEC.

```
% this scripts is to test the Unified Linear Input and State Estimator
% (ULISE algorithm) described in
%
% S. Z. Yong, M. Zhu, and E. Frazzoli,  A unified filter for simultaneous
% input and state estimation of linear discrete-time stochastic systems,
% Automatica, vol. 63, pp. 321 329 , 2016.
%
% Both Time-Varying and Steady-State versions are implemented. Pressure and
% acceleration measurements are used to estimate state and excitation force
%
% Sandia National Laboratories is a multi-mission laboratory managed and
% operated by National Technology and Engineering Solutions of Sandia,
% LLC., a wholly owned subsidiary of Honeywell International, Inc., for the
% U.S. Department of Energy?s National Nuclear Security Administration
% under contract DE-NA0003525.
%
% G. Bacelli, R. Coe
% Sandia National Laboratories
% 2017


clc
clear

% load identified parametric WEC model model
WEC = load('WEC_param_model_1DOF.mat'); % radiatiojn impedance model
Gr = struct2array(load('Gr_model.mat')); % radiation pressure model
Ge = struct2array(load('Ge_model.mat')); % excitation pressure model

Tc = 1e-3;
N = 5e4;

% load excitation force
Fe = load('Exc_time_series.mat'); %(excitation force in kN)
t = (1:N)*Tc;
d = interp1(Fe.t_trim, Fe.Fexc_td, t, 'pchip')*1e-3; % control input (PTO force in kN). Open loop, no control implemented

u = .5*sin(2*pi*0.75*t);

mass = 858.3987;
Ainf_hat = 822.3799;
K = 2.3981e+04;

Ar = WEC.rad_sys.a;
Br = WEC.rad_sys.b;
Cr = WEC.rad_sys.c;

Bf = WEC.B_eq_mat(9);

Am = [-Bf/(mass + Ainf_hat)     -K/(mass + Ainf_hat)         -Cr/(mass + Ainf_hat);
    1                    0                      zeros(1,length(Ar));
    Br                   zeros(length(Ar),1)    Ar];

% input matrix (1e3 factor is used to acccount for the forces that expressed are in kN)
Bm = 1e3*[1/(mass + Ainf_hat);
    0                     ;
    zeros(length(Ar),1)   ];
```

```
Cm = Am(1,:);

Dm = Bm(1);

Ai = blkdiag(Am, Gr.A);
Bi = [Bm; Gr.B];
Ci = blkdiag(Cm, Gr.C);
Di = [Dm; Gr.D];

ni = size(Ai,1);
ne = size(Ge.A,1);
Ac = [Ai,            Bi*Ge.C;
      zeros(ne, ni) Ge.A];

Bc = [Bi, Bi*Ge.D; zeros(ne,1), Ge.B];

C = [Ci, Di*Ge.C];
D = [Di, Di*Ge.D + [0;1] ];

sys_c = ss(Ac, Bc, C, D);
sys_d = c2d(sys_c, Tc);

Ge_d = c2d(Ge,Tc);

Ce = Ge_d.C;
De = Ge_d.D;

n = size(C,2);
p = size(C,1);

% convert continuous time model to discrete time
A = expm(Ac*Tc);
B = Ac\(A - eye(n))*Bc;

G = B(:,1);
B = B(:,1);

H = D(:,2);
D = D(:,1);

% process noise
w = .00001*(rand(N,ni+ne)-0.5);
% measurements noise
v = (rand(N,p)-0.5) * diag([.05, 0.05]);

Q = cov(w);
R = cov(v);

%% Time Varying filter

% initialize variables
P_x_k_k = 0.001*eye(n);
P_xd_k = 0.001*ones(n,1);
P_d_k = 0.001;

x = zeros(n,1);
x_k_k = x;
d_k = 0;

In = eye(n);
Ip = eye(p);

% preallocation
x_vec = zeros(n,N);
y_vec = zeros(p,N);
d_k_vec = zeros(N,1);
x_k_vec = zeros(n,N);
```

```
Fe_est = zeros(N,1);
Fe = Fe_est;

tic

for ii = 1:N

    x = A*x + B*u(ii) + G*d(ii) + w(ii,:)';
    y = C*x + D*u(ii) + H*d(ii) + v(ii,:)';

    Fe(ii) = Ce*x(ni+1:end) + De*d(ii);

    x_vec(:,ii) = x;
    y_vec(:,ii) = y;

    x_k_k1 = A*x_k_k + B*u(ii) + G*d_k;
    P_x_k_k1 = A*P_x_k_k*A' + G*P_xd_k'*A' + A*P_xd_k*G' + G*P_d_k*G' + Q;
    R_t_k = C*P_x_k_k1*C' + R;
    Kk = P_x_k_k1*C'/R_t_k;
    Lk = Kk*(Ip-((H/((H'/R_t_k)*H))*H'/R_t_k ));
    x_k_k = x_k_k1 + Lk*(y - C*x_k_k1 - D*u(ii) );
    P_x_k_k = (In-Lk*C)*P_x_k_k1*(In-Lk*C)' + Lk*R*Lk';
    R_ts_k = (Ip-C*Lk)*R_t_k*(Ip-C*Lk)';
    P_d_k = inv((H'/R_ts_k)*H);
    Mk = ((H'/R_ts_k)*H)\(H'/R_ts_k);
    d_k = Mk*(y - C*x_k_k - D*u(ii));
    P_xd_k = -P_x_k_k*C'*Mk' + Lk*R*Mk';

    d_k_vec(ii) = d_k;
    x_k_vec(:,ii) = x_k_k;
    Fe_est(ii) = Ce*x_k_k(ni+1:end) + De*d_k;

end

t_tv = toc;
disp('done TV')

%% Steady State filter

L_inf = Lk;
M_inf = Mk;

% preallocation
d_k_vec_inf = zeros(N,1);
x_vec_inf = zeros(n,N);
y_vec_inf = zeros(p,N);
x_vec_est_inf = x_vec_inf;
Fe_est_inf = zeros(N,1);

% initialization
x = zeros(n,1);
x_k_k_inf = zeros(n,1);
d_k1_inf = 0;

tic
for ii = 1:N

    x = A*x + B*u(ii) + G*d(ii) + w(ii,:)';
    y = C*x + D*u(ii) + H*d(ii) + v(ii,:)';

    x_vec_inf(:,ii) = x;
    y_vec_inf(:,ii) = y;

    x_k_k1_inf = A*x_k_k_inf + B*u(ii) + G*d_k1_inf;
    x_k_k_inf = x_k_k1_inf + L_inf*(y - C*x_k_k1_inf - D*u(ii) );
    d_k1_inf = M_inf*(y - C*x_k_k_inf - D*u(ii));

    d_k_vec_inf(ii) = d_k1_inf;
```

```
    x_vec_est_inf(:,ii) = x_k_k_inf;
    Fe_est_inf(ii) = Ce*x_k_k_inf(ni+1:end) + De*d_k1_inf;
end

t_ss = toc;
disp('done SS')

%% plotting

disp(['Time to compute Time-Varying filter: ' num2str(t_tv) 's'])
disp(['Time to compute Steady-State filter: ' num2str(t_ss) 's'])


figure(1)
subplot 211
plot(t, d_k_vec', t, d)
xlabel('time (s)')
ylabel('(kPa)')
grid on
legend({'$\hat{P}e$', '$Pe$' }, 'Interpreter', 'latex')
title('Time-Varying filter: Unknown input (Excitation pressure Pe)')

subplot 212
plot(t, Fe_est, t, Fe)
grid on
ylabel('(kN)')
xlabel('time (s)')
title('Excitation force')
legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')

figure(2)
subplot 211
plot(t, d_k_vec_inf', t, d)
xlabel('time (s)')
ylabel('(kPa)')
grid on
legend({'$\hat{P}e$', '$Pe$' }, 'Interpreter', 'latex')
title('Steady-State filter: Unknown input (Excitation pressure Pe)')

subplot 212
plot(t, Fe_est_inf, t, Fe)
grid on
ylabel('(kN)')
xlabel('time (s)')
title('Excitation force')
legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')

figure(3)
plot(t, d_k_vec - d_k_vec_inf)
grid on
xlabel('Time (s)')
title('Difference between Time-Varying and Steady-State filters')
legend('e_d')

figure(4)
subplot 211
plot(t, y_vec(1,:)')
xlabel('Time (s)')
ylabel('(m/s^2)')
title('Measured (noisy) Outputs')
legend({'$\ddot{z}$'}, 'Interpreter', 'latex')
grid on
subplot 212
plot(t, y_vec(2,:)')
xlabel('Time (s)')
ylabel('(kPa)', 'Interpreter', 'latex')
grid on
legend({'$P$'}, 'Interpreter', 'latex')
```

```
figure(5)
subplot 211
plot(t, x_vec(1,:)', t, x_k_vec(1,:)', t, x_vec_est_inf(1,:)')
grid on
xlabel('time (s)')
ylabel('(m/s)')
legend({'$v$', '$\hat{v}$', '$\hat{v}_{\infty}$'}, 'Interpreter', 'latex')
title('Estimated states')
grid on

subplot 212
plot(t, x_vec(2,:)', t, x_k_vec(2,:)', t, x_vec_est_inf(2,:)')
grid on
xlabel('time (s)')
ylabel('(m)')
legend({'$z$', '$\hat{z}$', '$\hat{z}_{\infty}$'}, 'Interpreter', 'latex')
grid on
```

State_and_unknown_input_estimator_pressure_acceleration.m

DISTRIBUTION:

1	MS 0899	Technical Library, 9536 (electronic copy)

Sandia National Laboratories