

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

1	Introduction	7
2	Using position	9
3	Using pressure	12
	References	15

Appendix

A	Sample code: position and acceleration measurements	17
B	Sample code: pressure and acceleration measurements	21

Figures

1	Block diagram of the buoy's dynamic model when position and acceleration are measured	11
2	Block diagram of the buoy's dynamic model when pressure and acceleration are measured.	13

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]

$$x_{k+1} = Ax_k + Bu_k + Gd_k + w_k \quad (1a)$$

$$y_k = Cx_k + Du_k + Hd_k + v_k, \quad (1b)$$

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

$$\dot{x}_i = A_i x_i + B_i (F_e + F_a) \quad (2a)$$

$$y_i = C_i x_i + D_i (F_e + F_a) \quad (2b)$$

where the state vector is

$$x = \begin{bmatrix} z \\ \dot{z} \\ x_r \end{bmatrix}, \quad (3)$$

and

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

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

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

$$\dot{x}_r = A_r x_r + B_r \dot{z} \quad (6a)$$

$$F_r = C_r x_r. \quad (6b)$$

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 $\omega \rightarrow \infty$. 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

$$A = e^{A_i T_c} \quad (7)$$

$$B = A_i^{-1} (A - I) B_i \quad (8)$$

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

$$C = C_i \quad D = D_i \quad (9)$$

$$G = B \quad H = D. \quad (10)$$

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:

- 1: $\hat{x}_{0|0} = \mathbb{E}[x_0]$
- 2: $\hat{d}_0 = H^\dagger (y_0 - C\hat{x}_{0|0} - Du_0)$
- 3: $P_{0|0}^x = \mathcal{P}_0^x$
- 4: $P_0^d = \mathcal{P}_0^d$
- 5: $P_0^{xd} = \mathcal{P}_0^{xd}$
- 6: $Q = \mathbb{E}[w w^T]$
- 7: $R = \mathbb{E}[v v^T]$

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

- 8: **for** $k = 1$ to N **do**
- ▷ One-Step prediction
- 9: $\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} + G\hat{d}_{k-1}$
- 10: $P_{k|k-1}^x = AP_{k-1|k-1}^x A^T + GP_{k-1}^{xdT} A^T + AP_{k-1}^{xd} G^T + GP_{k-1}^d G^T + Q$
- 11: $\tilde{R}_k = CP_{k|k-1}^x C^T$
- ▷ Measurements update
- 12: $K_k = P_{k|k-1}^x C^T \tilde{R}_k^{-1}$
- 13: $L_k = K_k (I - H (H^T \tilde{R}_k^{-1} H)^{-1} H^T \tilde{R}_k^{-1})$
- 14: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k (y_k - C\hat{x}_{k|k-1} - Du_k)$
- 15: $P_{k|k}^x = (I - L_k C) P_{k|k-1}^x (I - L_k C)^T + L_k R L_k^T$
- ▷ Estimation of unknown input
- 16: $\tilde{R}_k^* = (I - CL_k) \tilde{R}_k (I - CL_k)^T$
- 17: $P_k^d = (H^T \tilde{R}_k^{*-1} H)^{-1}$
- 18: $M_k = P_k^d H^T \tilde{R}_k^{*-1}$
- 19: $\hat{d}_k = M_k (y_k - C\hat{x}_{k|k} - Du_k)$
- 20: $P_k^{xd} = -P_{k|k}^x C^T M_k^T + L_k R M_k^T$
- 21: **end for**

Algorithm 2 Steady-State Unknown Input and State Estimator.

▷ Initialize:

- 1: $\hat{x}_{0|0} = \mathbb{E}[x_0]$
- 2: $\hat{d}_0 = H^\dagger (y_0 - C\hat{x}_{0|0} - Du_0)$
- 3: $L_\infty = \lim_{k \rightarrow \infty} L_k$
- 4: $M_\infty = \lim_{k \rightarrow \infty} M_k$

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

- 5: **for** $k = 1$ to N **do**
- 6: $\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} + G\hat{d}_{k-1}$ ▷ One-Step prediction
- 7: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_\infty (y_k - C\hat{x}_{k|k-1} - Du_k)$ ▷ State estimation
- 8: $\hat{d}_k = M_\infty (y_k - C\hat{x}_{k|k} - Du_k)$ ▷ Unknown input estimation
- 9: **end for**

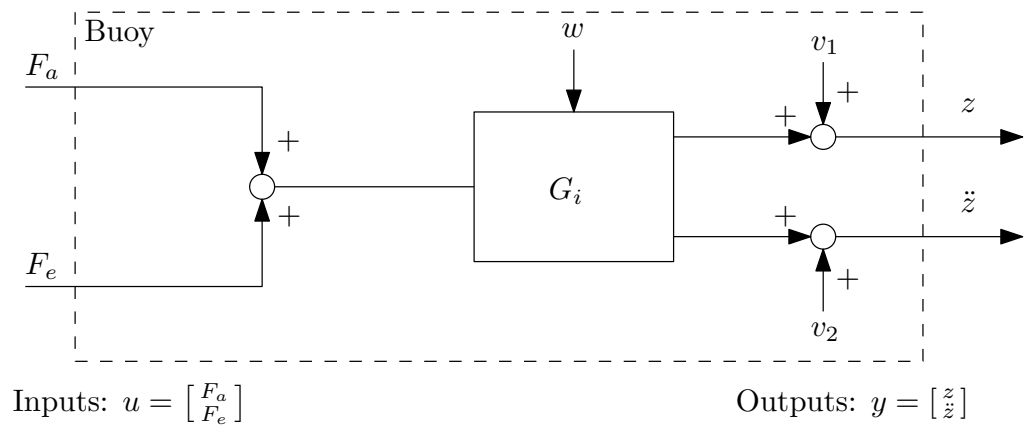


Figure 1. Block diagram of the buoy's dynamic model when position and acceleration are measured

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

$$\dot{x}_i = A_i x_i + B_i (F_e + F_a) \quad (11a)$$

$$y_i = C_i x_i + D_i (F_e + F_a) \quad (11b)$$

where the output vector y_i is

$$y_i = \begin{bmatrix} \ddot{z} \\ P_r \end{bmatrix}, \quad (12)$$

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

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

$$\dot{x}_e = A_e x_e + B_e P_e \quad (13a)$$

$$F_e = C_e x_e + D_e P_e. \quad (13b)$$

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

$$\dot{x} = A_c x + B_c \begin{bmatrix} F_a \\ P_e \end{bmatrix} \quad (14a)$$

$$y = C_c x + D_c \begin{bmatrix} F_a \\ P_e \end{bmatrix} \quad (14b)$$

where the state vector is

$$x = \begin{bmatrix} x_i \\ x_e \end{bmatrix} \quad (15)$$

and where the output vector is

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

The system matrices are

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

$$C_c = \begin{bmatrix} C_i & D_i C_e \end{bmatrix} \quad D_c = \begin{bmatrix} D_i & D_i D_e + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} \quad (18)$$

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

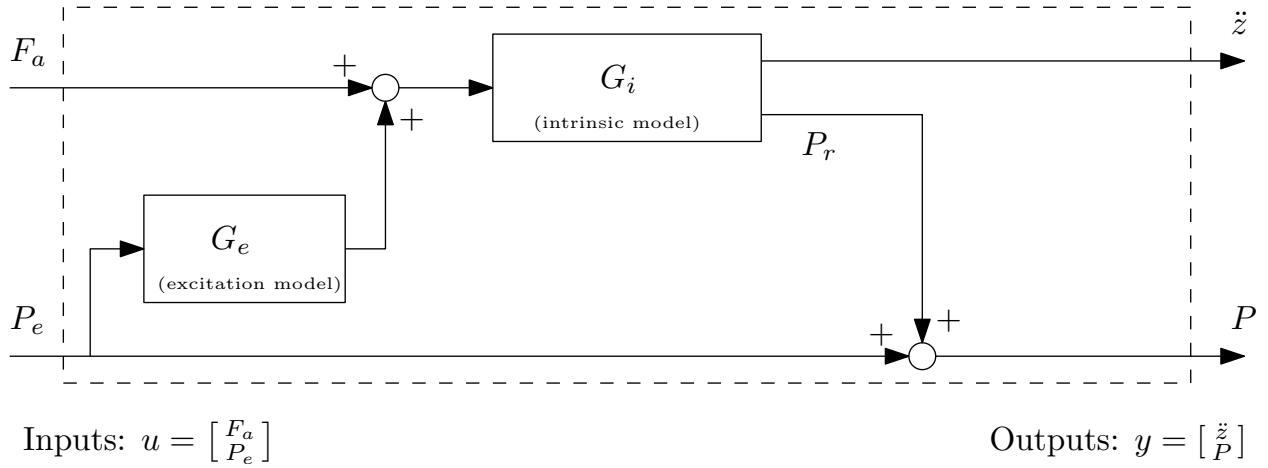


Figure 2. Block diagram of the buoy's dynamic model when pressure and acceleration are measured.

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

$$A = e^{A_c T_c} \quad (19)$$

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

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:

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

$$C = C_c \quad D_c = [D \quad H] \quad (22)$$

$$(23)$$

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:

- 1: $\hat{x}_{0|0} = \mathbb{E}[x_0]$
- 2: $\hat{d}_0 = H^\dagger (y_0 - C\hat{x}_{0|0} - Du_0)$
- 3: $P_{0|0}^x = \mathcal{P}_0^x$
- 4: $P_0^d = \mathcal{P}_0^d$
- 5: $P_0^{xd} = \mathcal{P}_0^{xd}$
- 6: $Q = \mathbb{E}[w w^T]$
- 7: $R = \mathbb{E}[v v^T]$

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

- 8: **for** $k = 1$ to N **do**
- ▷ One-Step prediction
- 9: $\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} + G\hat{d}_{k-1}$
- 10: $P_{k|k-1}^x = AP_{k-1|k-1}^x A^T + GP_{k-1}^{xd} A^T + AP_{k-1}^{xd} G^T + GP_{k-1}^d G^T + Q$
- 11: $\tilde{R}_k = CP_{k|k-1}^x C^T$
- ▷ Measurements update
- 12: $K_k = P_{k|k-1}^x C^T \tilde{R}_k^{-1}$
- 13: $L_k = K_k (I - H (H^T \tilde{R}_k^{-1} H)^{-1} H^T \tilde{R}_k^{-1})$
- 14: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k (y_k - C\hat{x}_{k|k-1} - Du_k)$
- 15: $P_{k|k}^x = (I - L_k C) P_{k|k-1}^x (I - L_k C)^T + L_k R L_k^T$
- ▷ Estimation of unknown input
- 16: $\tilde{R}_k^* = (I - CL_k) \tilde{R}_k (I - CL_k)^T$
- 17: $P_k^d = (H^T \tilde{R}_k^{*-1} H)^{-1}$
- 18: $M_k = P_k^d H^T \tilde{R}_k^{*-1}$
- 19: $\hat{d}_k = M_k (y_k - C\hat{x}_{k|k} - Du_k)$
- 20: $P_k^{xd} = -P_{k|k}^x C^T M_k^T + L_k R M_k^T$
- 21: **end for**

Algorithm 4 Steady-State Unknown Input and State Estimator.

▷ Initialize:

- 1: $\hat{x}_{0|0} = \mathbb{E}[x_0]$
- 2: $\hat{d}_0 = H^\dagger (y_0 - C\hat{x}_{0|0} - Du_0)$
- 3: $L_\infty = \lim_{k \rightarrow \infty} L_k$
- 4: $M_\infty = \lim_{k \rightarrow \infty} M_k$

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

- 5: **for** $k = 1$ to N **do**
- 6: $\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1} + G\hat{d}_{k-1}$ ▷ One-Step prediction
- 7: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_\infty (y_k - C\hat{x}_{k|k-1} - Du_k)$ ▷ State estimation
- 8: $\hat{d}_k = M_\infty (y_k - C\hat{x}_{k|k} - Du_k)$ ▷ Unknown input estimation
- 9: **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.

```
1 % this scripts is to test the Unified Linear Input and State Estimator
2 % (ULISE algorithm) described in
3 %
4 % S. Z. Yong, M. Zhu, and E. Frazzoli, A unified filter for simultaneous
5 % input and state estimation of linear discrete-time stochastic systems,
6 % Automatica, vol. 63, pp. 321 329, 2016.
7 %
8 % Both Time-Varying and Steady-State versions are implemented.
9 % Position and acceleration measurements are used to estimate state and
10 % excitation force
11 %
12 % Sandia National Laboratories is a multi-mission laboratory managed and
13 % operated by National Technology and Engineering Solutions of Sandia,
14 % LLC., a wholly owned subsidiary of Honeywell International, Inc., for the
15 % U.S. Department of Energy's National Nuclear Security Administration
16 % under contract DE-NA0003525.
17 %
18 % G. Bacelli, R. Coe
19 % Sandia National Laboratories
20 % 2017
21
22
23 clc
24 clear
25
26 % load identified parametric WEC model model
27 WEC = load('WEC_param_model_1DOF.mat');
28
29 Tc = 1e-3; % sampling time
30 N = 5e4; % number of simulation steps
31
32 % load excitation force and interploate
33 Fe = load('Exc_time_series.mat');
34 t = (1:N)*Tc;
35 d = interp1(Fe.t_trim, Fe.Fexc_td, t, 'pchip')*1e-3; %(excitation force in kN)
36
37 u = 0.5*sin(2*pi*0.75*t); % control input (PTO force in kN). Open loop, no control implemented
38
39 mass = 858.3987;
40 Ainf_hat = 822.3799;
41 K = 2.3981e+04;
42
43 Ar = WEC.rad_sys.a;
44 Br = WEC.rad_sys.b;
45 Cr = WEC.rad_sys.c;
46
47 Bf = WEC.B_eq_mat(9);
48
49 Ac = [-Bf/(mass + Ainf_hat)    -K/(mass + Ainf_hat)    -Cr/(mass + Ainf_hat);
50       1                        0                        zeros(1,length(Ar));
51       Br                        zeros(length(Ar),1)      Ar                ];
52
53 % input matrix (1e3 factor is used to account for the forces that expressed are in kN)
54 Bc = 1e3*[1/(mass + Ainf_hat);
55          0                        ;
56          zeros(length(Ar),1)];
57
58 % measurements: position and acceleration
```

```

59 C = [0 1 zeros(1,length(Ar));
      Ac(1,:)          ];
61
63 D = [0; Bc(1)];
65
67 n = size(C,2);
69 p = size(C,1);
71
73 % convert continuous time model to discrete time
75 A = expm(Ac*Tc);
77 B = Ac\ (A - eye(n))*Bc;
79
81 G = B;
83 H = D;
85
87 % process noise
89 w = .00001*(rand(N,4)-0.5);
91 % measurements noise
93 v = (rand(N,p)-0.5) * diag([.0001,0.05]);
95
97 Q = cov(w);
99 R = cov(v);
101
103 %% Time varying filter
105
107 % initialize variables
109 P_x_k_k = 0.001*eye(n);
111 P_xd_k = 0.001*ones(n,1);
113 P_d_k = 0.001;
115
117 x = zeros(n,1);
119 x_k_k = x;
121 d_k = 0;
123
125 In = eye(n);
127 Ip = eye(p);
129
131 % preallocation
133 d_k_vec = zeros(N,1);
135 x_k_vec = zeros(n,N);
137 x_vec = zeros(n,N);
139 y_vec = zeros(p,N);
141
143 tic
145
147 for ii = 1:N
149
151     x = A*x + B*u(ii) + G*d(ii) + w(ii,:);
153     y = C*x + D*u(ii) + H*d(ii) + v(ii,:);
155
157     x_vec(:,ii) = x;
159     y_vec(:,ii) = y;
161
163     x_k_k1 = A*x_k_k + B*u(ii) + G*d_k;
165     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;
167     R_t_k = C*P_x_k_k1*C' + R;
169     Kk = P_x_k_k1*C'/R_t_k;
171     Lk = Kk*(Ip - ((H'/(H'/R_t_k)*H))*H'/R_t_k);
173     x_k_k = x_k_k1 + Lk*(y - C*x_k_k1 - D*u(ii));
175     P_x_k_k = (In-Lk*C)*P_x_k_k1*(In-Lk*C)' + Lk*R*Lk';
177     R_ts_k = (Ip-C*Lk)*R_t_k*(Ip-C*Lk)';
179     P_d_k = inv((H'/R_ts_k)*H);
181     Mk = ((H'/R_ts_k)*H)\(H'/R_ts_k);
183     d_k = Mk*(y - C*x_k_k - D*u(ii));
185     P_xd_k = -P_x_k_k*C'*Mk' + Lk*R*Mk';
187
189     d_k_vec(ii) = d_k;

```

```

127     x_k_vec(:,ii) = x_k_k;
129 end
131 t_tv = toc;
132 disp('done TV')
133 %% steady state filter
135 L_inf = Lk;
137 M_inf = Mk;
139 % preallocation
140 d_k_vec_inf = zeros(N,1);
141 x_vec_inf = zeros(n,N);
142 y_vec_inf = zeros(p,N);
143 x_vec_est_inf = x_vec_inf;
145 % initialization
146 x = zeros(n,1);
147 x_k_k_inf = zeros(n,1);
148 d_kl_inf = 0;
149 tic
151 for ii = 1:N
153     x = A*x + B*u(ii) + G*d(ii) + w(ii,:);
154     y = C*x + D*u(ii) + H*d(ii) + v(ii,:);
155
156     x_vec_inf(:,ii) = x;
157     y_vec_inf(:,ii) = y;
159     x_k_kl_inf = A*x_k_k_inf + B*u(ii) + G*d_kl_inf;
160     x_k_k_inf = x_k_kl_inf + L_inf*(y - C*x_k_kl_inf - D*u(ii) );
161     d_kl_inf = M_inf*(y - C*x_k_k_inf - D*u(ii));
163     d_k_vec_inf(ii) = d_kl_inf;
164     x_vec_est_inf(:,ii) = x_k_k_inf;
165 end
167 t_ss = toc;
168 disp('done SS')
169
171 %% plotting
173 disp(['Time to compute Time-Varying filter: ' num2str(t_tv) 's'])
174 disp(['Time to compute Steady-State filter: ' num2str(t_ss) 's'])
175
176 figure(1)
177 plot(t, d_k_vec, t, d)
178 xlabel('time (s)')
179 ylabel('(kN)')
180 grid on
181 title('Time-Varying filter')
182 legend({'$\hat{F}e_{\infty}$', '$Fe$'}, 'Interpreter', 'latex')
183
184 figure(2)
185 plot(t, d_k_vec_inf, t, d)
186 xlabel('time (s)')
187 ylabel('(kN)')
188 grid on
189 title('Steady-State filter')
190 legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')
191
192 figure(3)
193 plot(t, d_k_vec - d_k_vec_inf)
194 grid on

```

```

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

199 figure(4)
    subplot 211
201 plot(t, y_vec(1,:))
    xlabel('Time (s)')
203 ylabel('(m)')
    title('Measured (noisy) Outputs')
205 legend({'$z$'}, 'Interpreter', 'latex')
    grid on
207 subplot 212
    plot(t, y_vec(2,:))
209 xlabel('Time (s)')
    ylabel('(m/s^2)')
211 legend({'$\ddot{z}$'}, 'Interpreter', 'latex')
    grid on
213

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

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

```

State_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.

```
1 % this scripts is to test the Unified Linear Input and State Estimator
2 % (ULISE algorithm) described in
3 %
4 % S. Z. Yong, M. Zhu, and E. Frazzoli, A unified filter for simultaneous
5 % input and state estimation of linear discrete-time stochastic systems,
6 % Automatica, vol. 63, pp. 321 329, 2016.
7 %
8 % Both Time-Varying and Steady-State versions are implemented. Pressure and
9 % acceleration measurements are used to estimate state and excitation force
10 %
11 % Sandia National Laboratories is a multi-mission laboratory managed and
12 % operated by National Technology and Engineering Solutions of Sandia,
13 % LLC., a wholly owned subsidiary of Honeywell International, Inc., for the
14 % U.S. Department of Energy's National Nuclear Security Administration
15 % under contract DE-NA0003525.
16 %
17 % G. Bacelli, R. Coe
18 % Sandia National Laboratories
19 % 2017
20
21
22
23
24
25 % load identified parametric WEC model model
26 WEC = load('WEC_param_model_1DOF.mat'); % radiatiojn impedance model
27 Gr = struct2array(load('Gr_model.mat')); % radiation pressure model
28 Ge = struct2array(load('Ge_model.mat')); % excitation pressure model
29
30 Tc = 1e-3;
31 N = 5e4;
32
33 % load excitation force
34 Fe = load('Exc_time_series.mat'); %(excitation force in kN)
35 t = (1:N)*Tc;
36 d = interp1(Fe.t_trim, Fe.Fexc_td, t, 'pchip')*1e-3; % control input (PTO force in kN). Open loop
37     , no control implemented
38
39 u = .5*sin(2*pi*0.75*t);
40
41 mass = 858.3987;
42 Ainf_hat = 822.3799;
43 K = 2.3981e+04;
44
45 Ar = WEC.rad_sys.a;
46 Br = WEC.rad_sys.b;
47 Cr = WEC.rad_sys.c;
48
49 Bf = WEC.B_eq_mat(9);
50
51 Am = [-Bf/(mass + Ainf_hat)    -K/(mass + Ainf_hat)    -Cr/(mass + Ainf_hat);
52       1                        0                      zeros(1,length(Ar));
53       Br                      zeros(length(Ar),1)    Ar];
54
55 % input matrix (1e3 factor is used to acccount for the forces that expressed are in kN)
56 Bm = 1e3*[1/(mass + Ainf_hat);
57           0                        ;
58           zeros(length(Ar),1)    ];
```

```

59 Cm = Am(1,:);

61 Dm = Bm(1);

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

67
ni = size(Ai,1);
69 ne = size(Ge.A,1);
Ac = [Ai,          Bi*Ge.C;
71      zeros(ne, ni) Ge.A];

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

75 C = [Ci, Di*Ge.C];
D = [Di, Di*Ge.D + [0;1] ];
77
sys_c = ss(Ac, Bc, C, D);
79 sys_d = c2d(sys_c, Tc);

81 Ge_d = c2d(Ge,Tc);

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

85
n = size(C,2);
87 p = size(C,1);

89 % convert continuous time model to discrete time
A = expm(Ac*Tc);
91 B = Ac\ (A - eye(n))*Bc;

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

95
H = D(:,2);
97 D = D(:,1);

99 % process noise
w = .00001*(rand(N,ni+ne)-0.5);
101 % measurements noise
v = (rand(N,p)-0.5) * diag([.05, 0.05]);

103
Q = cov(w);
105 R = cov(v);

107 %% Time Varying filter

109 % initialize variables
P_x_k_k = 0.001*eye(n);
111 P_xd_k = 0.001*ones(n,1);
P_d_k = 0.001;

113
x = zeros(n,1);
115 x_k_k = x;
d_k = 0;

117
In = eye(n);
119 Ip = eye(p);

121 % preallocation
x_vec = zeros(n,N);
123 y_vec = zeros(p,N);
d_k_vec = zeros(N,1);
125 x_k_vec = zeros(n,N);

```

```

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

129 tic

131 for ii = 1:N

133     x = A*x + B*u(ii) + G*d(ii) + w(ii,:);
134     y = C*x + D*u(ii) + H*d(ii) + v(ii,:);
135
136     Fe(ii) = Ce*x(ni+1:end) + De*d(ii);
137
138     x_vec(:,ii) = x;
139     y_vec(:,ii) = y;

141     x_k_k1 = A*x_k_k + B*u(ii) + G*d_k;
142     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;
143     R_t_k = C*P_x_k_k1*C' + R;
144     Kk = P_x_k_k1*C'/R_t_k;
145     Lk = Kk*(Ip-((H'/(H'/R_t_k)*H))*H'/R_t_k);
146     x_k_k = x_k_k1 + Lk*(y - C*x_k_k1 - D*u(ii));
147     P_x_k_k = (In-Lk*C)*P_x_k_k1*(In-Lk*C)' + Lk*R*Lk';
148     R_ts_k = (Ip-C*Lk)*R_t_k*(Ip-C*Lk)';
149     P_d_k = inv((H'/R_ts_k)*H);
150     Mk = ((H'/R_ts_k)*H)\(H'/R_ts_k);
151     d_k = Mk*(y - C*x_k_k - D*u(ii));
152     P_xd_k = -P_x_k_k*C'*Mk' + Lk*R*Mk';

153
154     d_k_vec(ii) = d_k;
155     x_k_vec(:,ii) = x_k_k;
156     Fe_est(ii) = Ce*x_k_k(ni+1:end) + De*d_k;

157 end

159 t_tv = toc;
161 disp('done TV')

163 %% Steady State filter

165 L_inf = Lk;
166 M_inf = Mk;
167

168 % preallocation
169 d_k_vec_inf = zeros(N,1);
170 x_vec_inf = zeros(n,N);
171 y_vec_inf = zeros(p,N);
172 x_vec_est_inf = x_vec_inf;
173 Fe_est_inf = zeros(N,1);

175 % initialization
176 x = zeros(n,1);
177 x_k_k_inf = zeros(n,1);
178 d_k1_inf = 0;

179 tic
181 for ii = 1:N

183     x = A*x + B*u(ii) + G*d(ii) + w(ii,:);
184     y = C*x + D*u(ii) + H*d(ii) + v(ii,:);
185
186     x_vec_inf(:,ii) = x;
187     y_vec_inf(:,ii) = y;

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

193     d_k_vec_inf(ii) = d_k1_inf;

```

```

195     x_vec_est_inf(:,ii) = x_k_k_inf;
196     Fe_est_inf(ii) = Ce*x_k_k_inf(ni+1:end) + De*d_k1_inf;
197 end
198
199 t_ss = toc;
200 disp('done SS')
201 %% plotting
202
203 disp(['Time to compute Time-Varying filter: ' num2str(t_tv) 's'])
204 disp(['Time to compute Steady-State filter: ' num2str(t_ss) 's'])
205
206
207 figure(1)
208 subplot 211
209 plot(t, d_k_vec', t, d)
210 xlabel('time (s)')
211 ylabel(' (kPa)')
212 grid on
213 legend({'$\hat{P}e$', '$Pe$'}, 'Interpreter', 'latex')
214 title('Time-Varying filter: Unknown input (Excitation pressure Pe)')
215
216 subplot 212
217 plot(t, Fe_est, t, Fe)
218 grid on
219 ylabel(' (kN)')
220 xlabel('time (s)')
221 title('Excitation force')
222 legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')
223
224 figure(2)
225 subplot 211
226 plot(t, d_k_vec_inf', t, d)
227 xlabel('time (s)')
228 ylabel(' (kPa)')
229 grid on
230 legend({'$\hat{P}e$', '$Pe$'}, 'Interpreter', 'latex')
231 title('Steady-State filter: Unknown input (Excitation pressure Pe)')
232
233 subplot 212
234 plot(t, Fe_est_inf, t, Fe)
235 grid on
236 ylabel(' (kN)')
237 xlabel('time (s)')
238 title('Excitation force')
239 legend({'$\hat{F}e$', '$Fe$'}, 'Interpreter', 'latex')
240
241 figure(3)
242 plot(t, d_k_vec - d_k_vec_inf)
243 grid on
244 xlabel('Time (s)')
245 title('Difference between Time-Varying and Steady-State filters')
246 legend('e_d')
247
248 figure(4)
249 subplot 211
250 plot(t, y_vec(1,:))
251 xlabel('Time (s)')
252 ylabel(' (m/s^2)')
253 title('Measured (noisy) Outputs')
254 legend({'$\ddot{z}$'}, 'Interpreter', 'latex')
255 grid on
256 subplot 212
257 plot(t, y_vec(2,:))
258 xlabel('Time (s)')
259 ylabel(' (kPa)', 'Interpreter', 'latex')
260 grid on
261 legend({'$P$'}, 'Interpreter', 'latex')

```



```

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

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

```

State_and_unknown_input_estimator_pressure_acceleration.m

DISTRIBUTION:

1 MS 0899 Technical Library, 9536 (electronic copy)

