

A Process Control System for the John F. Kennedy Memorial Bridge

Principal Investigator: Professor Pablo L. Durango-Cohen

A final report submitted to the Infrastructure Technology Institute

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

A Process Control System for the John F. Kennedy Memorial Bridge

Yikai Chen

David J. Corr

Pablo L. Durango-Cohen

Department of Civil & Environmental Engineering and Northwestern University Transportation Center, 2145 Sheridan Road, A332, Evanston, IL, 60208-3109, Phone: 847-491-4008, Fax: 847-491-4011, Email: pdc@northwestern.edu .

1. Background and Objective

The John F. Kennedy Memorial Bridge is a cantilever through truss that carries Interstate 65 across the Ohio River between Louisville, KY and Jeffersonville, IN. During a routine inspection in 2006, one of the anchor bolts on its northwest bearing was found fractured, leading to concerns over performances of the uplifting bearing. A retrofit system was then installed consisting of a threaded rod and clamp-down keepers. Additionally, continuous remote monitoring was implemented to provide thorough characterization of the retrofit performance in the long term, so as to support managerial decisions related to inspection & maintenance planning.

This system has continuously monitored the condition of the repaired bearing assembly, and was able to detect an abrupt failure in one of the components of the retrofit in the fall of 2008. This failure resulted in immediate and obvious changes to the stream of data collected by the system. However, there is a need for techniques to monitor long-term trends in condition, which may be subtle and not immediately obvious upon reviewing the raw data.

In this study, we developed and implemented an automatic system to process and analyze the data collected. The **objectives** are to:

- Allow engineers to make inferences about the structural integrity, i.e., the condition, of the elements, and how they evolve over time in response to normal operating factors such as weather, traffic loading, etc.
- Detect external events that may have either a transitory or a permanent effect on the structural integrity of the bridge elements and how they evolve over time.

In both cases the goal is to support decisions to inspect the components when structural deficiencies are either anticipated due to regular deterioration or directly related to external events.

During the progress of this project, the system has monitored and controlled the performance of the J.F.K memorial bridge using available data dating from summer 2008, and has demonstrated its capability to achieve the objectives listed above. However, due to unexpected breaks of the instrumentation process in early 2009 and after mid-2010, no complete data were available anymore to further testify and improve updated features of the system. This is beyond our control and we have started to implement the same set of techniques and framework in the system on other infrastructures for continued research.

In line with the USDOT's research goals, this work involves development and implementation of cutting-edge, transformative research tools to support information management, and decisions related to the management/renewal of surface transportation infrastructure. In addition, this work is also complementary to the work/expertise of Northwestern University's *Infrastructure Technology Institute* (ITI) in developing advanced remote monitoring systems.

A 3D perspective diagram of a mechanical assembly. It features a central grey block with a green hexagonal feature on its top surface. This block is supported by two vertical blue pillars. Two green nuts are positioned on the top surface, one on each side of the central block. Red arrows indicate the direction of strain: one arrow points horizontally away from the central block towards the left nut, labeled 'Axial strain'; another arrow points horizontally away from the central block towards the right nut, labeled 'Axial, torsion, bending strains'.

and related measurement noises. Even if tightly coupled in the data stream, these two sets of information provide different implications for managers/engineers, motivating them to focus on different perspectives of the process when trying to identify existing or potential issues.

Therefore, a **key feature** of the system developed herein is to distinguish measurement noises from system randomness. Such an idea could be formulated using a basic statistical model called “Random Walk” introduced in Durbin and Koopman (2001):

$$\text{Measurement}_t = \text{Condition}_t + \text{Error}_t \quad (1)$$

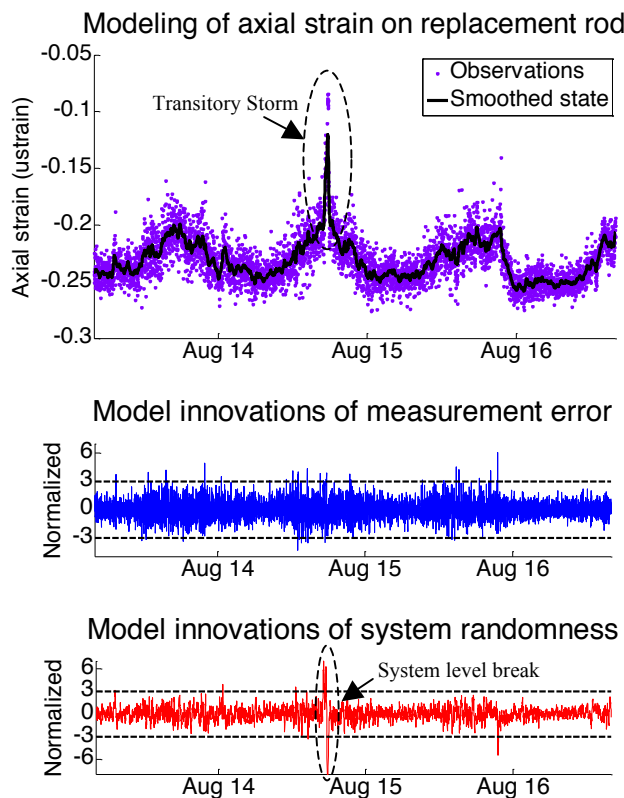
$$\text{Condition}_{t+1} = \text{Condition}_t + \text{Random}_t \quad (2)$$

At each time t , the observed data is an addition of the true facility condition and an error term in the measurement process, while the true facility condition transitions into the next time period with a slight random shift. It implements time series framework to fit the data and yields two paired sets of innovations: one illustrates errors in the measurement process while the other profiles randomness on the system level.

If the underlying facility is working properly, both the error term and the randomness term should appear as a Gaussian noise. As a result, according to the Central Limit Theorem, 99% of the data points should fall into a ± 3 standard deviation interval from the mean value. Points falling out of the interval would indicate an out-of-control occurrence.

The result can help management personnel to distinguish system level change in the structure from instrumentation defects, to identify explanatory causes, hence facilitate their decision making: whether to send a bridge inspector or an instrument technician.

Case I: Alerting Structural Level Break



In this case we implemented the aforementioned model to the axial strain of the replacement rod. The raw measurement observations were dotted in purple and the estimated true condition was curved in black.

In addition, the measurement errors and system randomness were normalized and plotted in separate frames, with ± 3 standard deviations drawn respectively as confidence boundaries. In our system, three consecutive points out of the boundaries would trigger an out-of-control event.

As a result, an alert of system level break was triggered on August 14th, which showed transitory effect on the structural performance of the bridge.

This event was later confirmed to correspond with a short-term storm.

Another key component of the system is to detect long-term trends in the structure performances, which is hardly available from short-term tests.

We developed time series models in a state space framework to exploit the measurement data, which is proved to be a statistically rigorous approach to estimating/forecasting facility conditions. Specifically there are two classes of models in this practice:

- i) Autoregressive Moving Average (ARMA) models
- ii) Structural time series models

Both of them have advantages and disadvantages depending on the study objective and properties of data. In this project where long-term measurements are to be recorded, it is of huge value to identify components such as trend, seasonal variation, and system randomness from the observations, and therefore we focused on structural time series models first.

A basic form of the structural time series model is formulated as follows:

$$\text{Measurement}_t = \text{Trend}_t + \text{Seasonal}_t + \text{Error}_t \quad (3)$$

$$\text{Trend}_{t+1} = \text{Trend}_t + \text{Slope}_t + \text{Random}_t^T \quad (4)$$

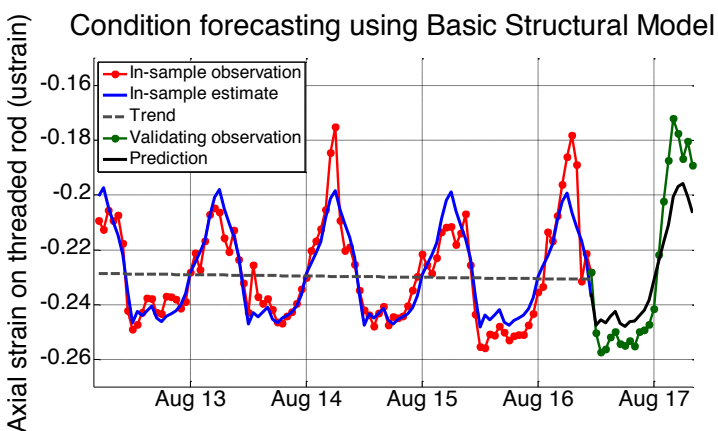
$$\text{Slope}_{t+1} = \text{Slope}_t + \text{Random}_t^L \quad (5)$$

$$\sum_{i=0}^{l-1} \text{Seasonal}_{t-i} = \text{Random}_t^S \quad (6)$$

At each time t , the observed data is an additive composition of trend, seasonal effect, and an error term, all of which follow a stochastic transitioning process. Random_t^T , Random_t^L , and Random_t^S are independent Gaussian noises and l is the cycle length of the seasonal effect. The resulting trend gives engineers reliable inferences about how the structure integrity evolves over time.

Case II: Detect Long-term Trends

In this case, we implemented the basic structure model on the measured axial strain of the replacement rod based on the data in August 2008. Four days of hourly observations were used to determine parameters of the model. And a fifth day was used to validate the results.



Observations were decomposed into different structural components, each showing a specific feature of the rod performance.

Importantly, a downward trend was detected (dashed grey) which indicated an increasing stress loaded on the rod. This was believed to have gradually and

latently contributed to the sudden break of the rod a month later.

4. Conclusion

We developed an automated system in this work to monitor and control the long-term deterioration process of the John F. Kennedy memorial bridge. It implemented statistical time series models in a state-space framework to make reliable inferences on the structural integrity of the facilities. Specifically, the system demonstrated its capability to:

- 1) Monitor the deterioration trend of the facilities and their seasonal patterns,
- 2) Detect burst events of the bearing system and make alerts to minimize risks,
- 3) Make predictions of its future condition with certain confidences, and hence
- 4) Provide valuable information for maintenance and repair decisions.

Further improvement of the system behavior and exploration of potential capabilities are unavailable at this point due to unexpected break of data stream caused by errors in the instrumentation process. However, we are continuing our researches onto other bridge infrastructures in a similar context with the goal to develop more advanced systems.

Throughout this project we disseminated the results of our work at academic conferences and research seminars. In addition, all the aforementioned models are formulated with computer programs written in Matlab. A list of all the deliverables are appended at the end of this report.

5. References

- 1) Ben-Akiva, M. and Ramaswamy, R. (1993), "An Approach for Predicting Latent Infrastructure Facility Deterioration", *Transportation Science*, 27(2), pp. 174-193.
- 2) Chu, C. and Durango-Cohen (2007), "Estimation of infrastructure performance models using state-space specifications of time series models", *Transportation Research Part C: Emerging Technologies*, 15 (1), pp. 17-32.
- 3) Humplick, F. (1992), "Highway Pavement Distress Evaluation: Modeling Measurement Error", *Transportation Research Part B: Methodological*, 26(2), pp. 135-154.
- 4) Durbin, J. and Koopman S.J. (2001), "Time Series Analysis by State Space Methods", *Oxford University Press*, New York, NY.

Appendix I – Related Presentations

1. Corr, D. and Durango-Cohen, "Exploring Advanced Inspection Technologies to Support Condition Assessment, Forecasting and Decision Making: A Process Control System for the John F. Kennedy Memorial Bridge", *Developing a Research Agenda for Transportation Infrastructure Preservation and Renewal, TRB Final Program*, November 12-13, 2009
2. Durango-Cohen, Working Seminar at the University of Minnesota, August 2010

Appendix II – Matlab Programs

1. Random Walk and Kalman Filter

```
%% Random Walk Model
% (1)  y = U + epsilon
% (2)  U(t+1) = U + eta

% U - the true state value
% y - n point observations
% P - state variance, P(t+1) = Var[U(t+1) | y(1~t)]

% R - variance of measurement errors, Var[epsilon]
% Q - variance of system randomness, Var[eta]

% x - estimation of U, x(t+1) = E[U(t+1) | y(1~t)]
% v - estimation error (also called 'innovations'), v = y - x
% F - prediction variance, F(t+1) = Var[v(t) | y(1~n)] = P(t+1) + R

% m - backward smoothing of U, m(t) = E[U(t) | y(1~n)]
% V - smoothing variance, V(t) = Var[U(t) | y(1~n)]
% e - smoothing error, e = E[epsilon | y(1~n)] = y - m
% r - (auxiliary) smoothing cumulant (weighted sum of innovations after t-1)
% N - (auxiliary) smoothing variance cumulant (weighted sum of inverse variances of innovations after t-1)

% sd_ep - standardised observation residuals, E[epsilon] / sqrt(Var[epsilon])
% sd_et - standardised state residuals, E[eta] / sqrt(Var[eta])
% u - (auxiliary) smoothing error, u = e/R = v/R - K*r
% D - (auxiliary) D = 1/F + K^2 * N

%% Loading Data
load set1.mat;
tpdata = ne_axial1(690:690+5041); len = 5042;

s.A = 1; % system transformation matrix
s.H = 1; % measurement transformation matrix
s.B = 0; % input transformation matrix
s.u = 0; % input control

R = 15099; % variance of measurement errors
Q = 1469.1; % variance of system error

s.x = tpdata(1); % initial state estimate
s.P = R; % initial state variance
s.F = s.P + R; % initial prediction variance
s.K = 0; % regression coefficient K = P/F = P/(P+R)
s.v = 0; % forecast error v = y-x

%% Calculate P(state variance) & K which don't depend on observations
for t=1:len
    s(end).y = tpdata(t);

    if isnan(s(end).y)    s(end).K = 0;    % set the current K to 0 if the observation is missing
    else    s(end).K = s(end).P / (s(end).P + R);    % otherwise update K = P/(P+R)
    end

    s(end+1).P = s(end).P * (1 - s(end).K) + Q;    % update P = P*(1-K) + Q
    s(end).F = s(end).P + R;    % prediction variance F = P + R
end

%% Use Kalman filter to do one-step filtering, i.e., x = E[U(t+1) | y(1~t)]
for t=1:len
    if isnan(s(t).y)    s(t).v = 0;    % set the current v to 0 if the observation is missing
```

```

else      s(t).v = s(t).y - s(t).x;      % otherwise update v = y - x, v is prediction error
end

s(t+1).x = s(t).x + s(t).K * s(t).v;      % update x = x + K*v

% calculate upper and lower bound of 50% CI on x
s(t).x_upper = s(t).x + 0.5*sqrt(s(t).F);
s(t).x_lower = s(t).x - 0.5*sqrt(s(t).F);
end

%% Backward smoothing after obtaining all observations, i.e., m = E[U(t) | y(1~n)]
s(len).r = 0;      % r is smoothing cumulant (weighted sum of innovations after t-1)
s(len).N = 0;      % N is smoothing variance cumulant (weighted sum of inverse variances of
innovations after t-1)
for t=1:len
    temp_r = s(len+1-t).v / s(len+1-t).F + (1-s(len+1-t).K)*s(len+1-t).r;      % backward update
r(t-1) = v/(P+R) + (1-K)*r
    temp_N = s(len+1-t).K/s(len+1-t).P + (1-s(len+1-t).K)^2 * s(len+1-t).N;      % backward update
N(t-1) = K/P + (1-K)^2 * N
    if t<len      % to avoid assign r(0) or N(0) which is illegal in Matlab
        s(len-t).r = temp_r;
        s(len-t).N = temp_N;
    end

    s(len+1-t).m = s(len+1-t).x + s(len+1-t).P*temp_r;      % calculate smoothed state m = x'hat' =
x + P*r(t-1)
    s(len+1-t).V = s(len+1-t).P - (s(len+1-t).P)^2*temp_N;      % calculate smoothed state variance
V = P - P^2 * N(t-1)
    s(len+1-t).e = s(len+1-t).y - s(len+1-t).m;      % calculate smoothed error e = y - m =
E[epsilon | y(1~n)];

    % calculate upper and lower bound of 99% CI on m
    s(len+1-t).m_upper = s(len+1-t).m + 3*sqrt( s(len+1-t).V );
    s(len+1-t).m_lower = s(len+1-t).m - 3*sqrt( s(len+1-t).V );
end

%% Smoothing disturbances: epsilon & eta
% they could be estimated from e=y-m, but sometimes it is needed to
% estimate them directly w/o calculating m (e.g. outlier detection)
for t=1:len
    s(t).u = s(t).v/s(t).F - s(t).K*s(t).r;
    s(t).D = 1/s(t).F + (s(t).K)^2*s(t).N;

    s(t).sd_ep = s(t).u / sqrt(s(t).D); % standardised observation residuals, E[epsilon] /
sqrt(Var[epsilon])
    s(t).sd_et = s(t).r / sqrt(s(t).N); % standardised state residuals, E[eta] / sqrt(Var[eta])
end

%% J-step minimum mean square error forecasting, i.e., E[y(n+j) | y(1~n)]
J=0;      % set to 0 to disable forecasting
for t=1:J
    s(len+t).x = s(len+1).x;      % E[y(n+j) | y(1~n)] = E[U(n+j) | y(1~n)] = x(n+1)
    s(len+t).P = s(len+1).P + (t-1)*Q;
    s(len+t).F = s(len+t).P + R;      % forecasting variance

    % upper and lower bound of 50% CI on forecast
    s(len+t).x_upper = s(len+t).x + 0.5*sqrt(s(len+t).F);
    s(len+t).x_lower = s(len+t).x - 0.5*sqrt(s(len+t).F);
end

%% Plot filtering and backward smoothing results
figure;
hold on;
scale = 0.031; % scale of conversion from mV to ustrain

% plot 99% CI of smoothed states
%aa = [1:len+J]; aa=[aa fliplr(aa)];

```

```

%bb = [[s(1:len+J).m_upper] fliplr([s(1:len+J).m_lower])];
%hf = fill(aa,bb, [0.75,0.75,0.75]);
%alpha(0.4); % set scale of transparency of the filled color

hz = plot([s(1:len).y].*scale,'g','MarkerSize',15, 'Color', [0.5,0,1]); % plot measurement data
hb = plot([s(1:len).m].*scale,'k','LineWidth',5); % plot (backward) smoothed states
h = gca; set(h,'FontSize',32);
xlim([1,len]); % limit of x axis
ylabel('Axial strain (ustrain)', 'FontSize', 32);
ylabel('Bending strain on the threaded rod (ft-lbs)', 'FontSize', 28);
set(gca,'xlim',[0 5040]);
set(gca,'xtick',[1200:1440:4080]);
set(gca,'xticklabel',{'Aug 14'; 'Aug 15'; 'Aug 16';});
legend1 = legend([hz,hb], 'Observations','Smoothed state','Location','NorthEast');
set(legend1, 'FontSize', 32);
text(800,-3.2,'System level break','FontSize',28);
title('Modeling of axial strain on replacement rod', 'FontSize', 40);
hold off;

%% Plot smoothed residuals for outlier detection
figure;
axes('position', [0.13, 0.55, 0.75,0.3]);
hold on;
plot([s(1:len).sd_ep],'b','LineWidth',2);
plot(3*ones(len,1),'k--','LineWidth',3);
plot(-3*ones(len,1),'k--','LineWidth',3);
h = gca; set(h,'FontSize',32);
xlim([1, len]);
ylim([-5,7]);
set(gca,'ytick',[-3 0 3 6]);
set(gca,'xlim',[0 5040]);
set(gca,'xtick',[1200:1440:4080]);
set(gca,'xticklabel',{'Aug 14'; 'Aug 15'; 'Aug 16';});
title('Model innovations of measurement error', 'FontSize', 40);
ylabel('Normalized')
hold off;

figure();
axes('position', [0.13, 0.08, 0.75, 0.3]);
hold on;
plot([s(1:len).sd_et],'r','LineWidth',2);
plot(3*ones(len,1),'k--','LineWidth',3);
plot(-3*ones(len,1),'k--','LineWidth',3);
h = gca; set(h,'FontSize',32);
xlim([1, len]);
ylim([-8,7]);
set(gca,'ytick',[-6 -3 0 3 6]);
set(gca,'xlim',[0 5040]);
set(gca,'xtick',[1200:1440:4080]);
set(gca,'xticklabel',{'Aug 14'; 'Aug 15'; 'Aug 16';});
%annotation('arrow',[0.4285 0.4576],[0.3508 0.3291],'LineWidth',3);
%text(540,6.1,'System level break','FontSize',28);
title('Model innovations of system randomness', 'FontSize', 40);
ylabel('Normalized')
hold off;

%% Plot ACF
figure();
axes('position', [0.13, 0.60, 0.4,0.3]);
hold on;
[acf1 lags1 bounds1] = autocorr([s(1:len).sd_ep]);

plot(lags1, acf1, 'r.', 'MarkerSize', 20);
line([lags1;lags1], [zeros(1,21);acf1], 'Color', 'r', 'LineWidth', 1);
line([0 20],[0 0], 'Color', 'k', 'LineWidth',1);
line([0.5 0.5; 20 20],[bounds1; bounds1], 'Color', 'b', 'Linewidth', 1);
h = gca; set(h,'FontSize',32);
title('ACF of measurement residual', 'FontSize', 40);
ylim([-0.2,1]);
set(gca,'ytick',[0,0.5,1]);

```

2. Basic Structural Models

```

ms = idss(A,B,C,D);
setstruc(ms,As,Bs,Cs,Ds,Ks,X0s);
set(ms, 'Ts', 1);

% estimate the model parameter using only sample 1:103
model = pem(data,ms);

% introduce the whole data set
time = 1:1:size(wholedata,1);
data1 = iddata([wholedata], zeros(size(wholedata,1),1), 1);
yp = predict(model,data1,21);
[yh,fit,x0] = compare(data, model);
mdl = yh{1}(:,1,:);

ttd = meanf(mdl.y(2:97), 24);
a = (ttd(4) - ttd(1))/72;
b = ttd(1) - 14*a;
trend = a * time(1:103) + b;

figure(); hold on; grid on;
plot(time(1:103),tpdata.*0.031,'r.-','MarkerSize',30,'LineWidth',3);
plot(time(1:103),mdl.y.*0.031,'b','LineWidth',4);
plot(time(1:103),trend.*0.031, 'm--','LineWidth',4,);
plot(time(103:124),wholedata(103:124).*0.031,'g.-','MarkerSize',30,'LineWidth',3)
plot(time(103:124), yp.y(103:124).*0.031,'k','LineWidth',4);
h = gca; set(h,'FontSize',28);
legend1 = legend('In-sample observation','In-sample estimate', 'Trend','Validating observation',
'Prediction');
set(legend1,'FontSize',22, 'Location','Northwest');
set(gca,'ylim',[-0.27 -0.15]);
set(gca,'xlim',[0 124]);
set(gca,'xtick',[20:24:118]);
set(gca,'xticklabel',['Aug 13'; 'Aug 14'; 'Aug 15'; 'Aug 16'; 'Aug 17']);
ylabel('Axial strain on threaded rod (ustrain)', 'FontSize', 32);
title('Condition forecasting using Basic Structural Model', 'FontSize', 36);
hold off;

%% Calculate RMSE
count = 0;
for i=103:124
    count = count + (yp.y(i)-wholedata(i)).^2;
end;
count = sqrt(count);

```