Aerodynamic System Identification
MAE 6243 Project

Charles O'Neill

December 17, 2004

1 Introduction

This project investigates aerodynamic system identification. An airfoil produces time varying loads based on the current and past boundary conditions; this project seeks to identify the lift loads resulting from pitch motion.

The first part discusses some unsteady aerodynamic theory. The second part discusses the identification process. The final part discusses conclusions and observations. Aerodynamic nomenclature is given after the reference section.

2 Unsteady Airfoil Theory

A classic unsteady aerodynamic theory is the incompressible, inviscid, flat plate, harmonic motion Theodorsen Problem. This project uses a reduced form of the Theodorsen result based on simplified geometry[1] and motion.[2] A survey of the method is found in Hodges and Pierce. The Theodorsen relationship between harmonic lift L and harmonic translational h and angular motions α is:

$$L = \pi\rho b^2 \left(\ddot{h} + U\dot{\alpha} - ba\ddot{\alpha}\right) + 2\pi b\rho U \left(\dot{h} + U\alpha + b\dot{\alpha}\left(\frac{1}{2} - a\right)\right)$$

where $C(k)$ is a complex valued function of Hankel functions

$$C(k) = \frac{H_1^{(2)}(k)}{H_1^{(2)}(k) + iH_0^{(2)}(k)}$$

The Theodorsen function C(k) is plotted in Figure 1 for reduced frequencies from 0 up to 10. The important concept to discover is that low frequencies approach the steady state lift, $C(0) \approx 1$. For high frequencies, C(k) approaches 0.5.

[1] Pitch axis is about the midchord

[2] Motion is only angular pitch motion.

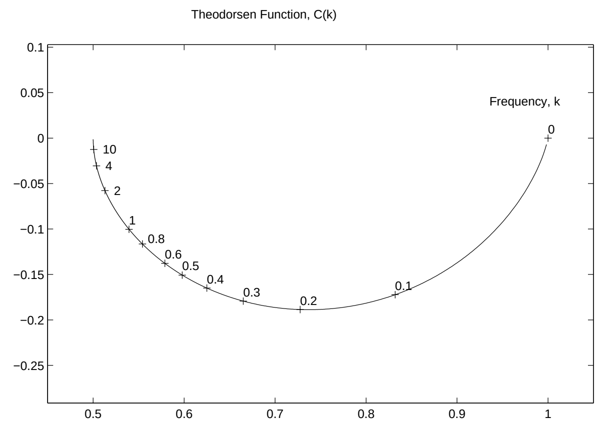

Figure 1: Theodorsen Function C(k)

For this project, simplified geometry and only the α degree of freedom reduce Theodorsen to a transfer function G. Lift is non-dimensionalized by the steady state lift qSC_{L_α}. The transfer function G is:

$$G(\omega) = C(k) + i\omega\frac{c}{4U}\left(1 + C(k)\right)$$

The Bode plot for G for the flight conditions used in the following system identification is given in Figure 2. The `theo.m` Matlab function generated the data. Three distinct areas are visible. First, a steady state lift occurs below

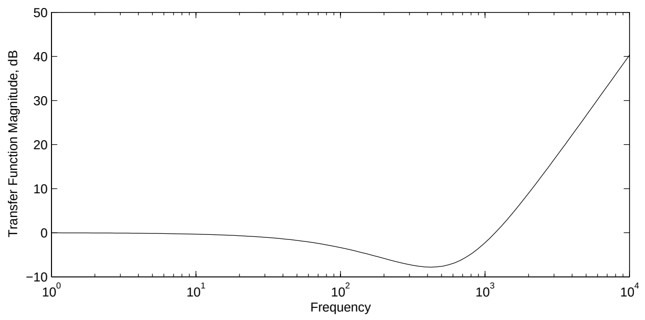

Figure 2: Theodorsen Lift

about about 10 Hertz. Between 10 and 500 Hz, lift magnitude decreases. Finally, above 500 Hz, the non-circulatory (ie. impulsive loading with C(k)=0.5) lift dominates. This section has an increase of approximately 40 dB per decade.

3 Aerodynamics

3.1 Fluid

A non-inertial, inviscid, finite element Computational Fluid Dynamics (CFD) program solves for the flow properties. The airfoil geometry is an NACA 0010 with a Mach 0.50 steady state flow solution shown in Figure 3.

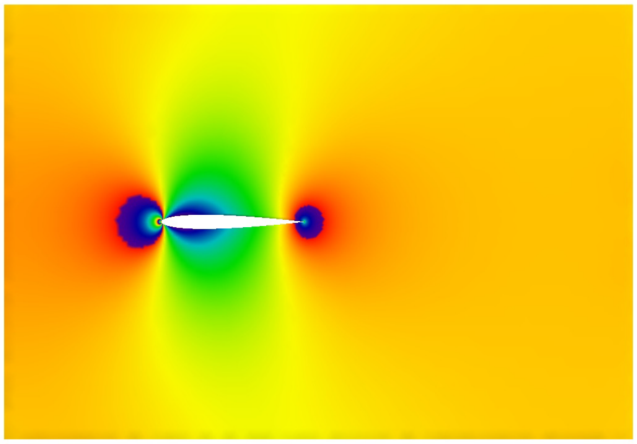

Figure 3: NACA 0010 Steady State Flow Solution

The CFD solver requires time-accurate displacement and velocity boundary conditions to maintain overall time-accuracy in the flow solution. Additionally, the CFD solver requires a relatively small solution timescale (0.2 ms) compared with the flow unsteadiness timescale.

3.2 Structure

Motion input is specified through a single time varying pitch boundary condition about the mid-chord. The training signal is input from force to displacement through a 2nd order mechanical system. Thus, all input signals are guaranteed two integrations for a total of -40 dB per decade. A negative side effect of this motion input is that the displacement has an increasing variance in time.

For this project, the input u(t) will be the angular displacement α; the output y(t) will be the lift force L. CFD data output is through the `xn.dat` data file shown in Figure 4. The input training signal was generated with `generatefrc.m`.

4 System Model

This projects investigates two signals y(t) and u(t) generated with an CFD solver. The objective is to analyze and develop a linear model with system

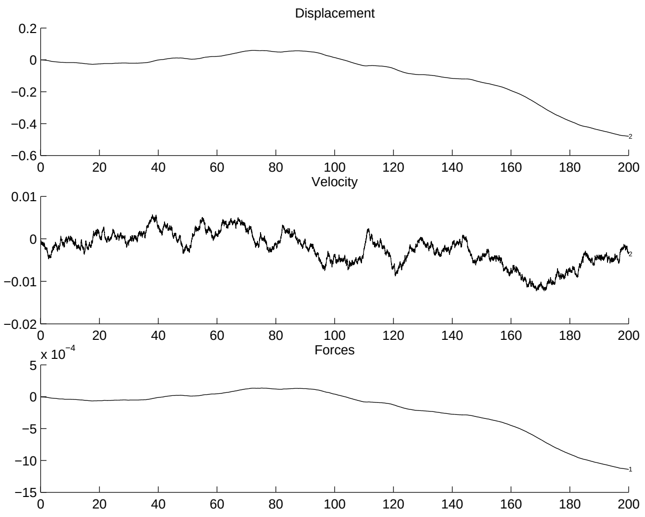

Figure 4: CFD data output `xn.dat`

identification techniques.

The system model will be the Box-Jenkins form:

$$y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$$

with the following one step ahead predictor:

$$\hat{y}(t) = \frac{D(q)B(q)}{C(q)F(q)}u(t) + \frac{C(q) - D(q)}{C(q)}y(t)$$

5 Preliminary Identification

The objective of this section is to estimate time scales and model orders.

5.1 Time Scales

The Theodorsen theory allows for an initial time scale estimate. As discussed above, resampling the CFD data is likely to be needed. Figure 2 shows that above approximately 1000 Hz is dominated by a *zero* type behavior, so resampling to below 10000 Hz appears reasonable. In practice, the actual resampling rate (1:10) was determined through experimentation and model order tests. The sampling occurs in the `orderest.m` Matlab script with the `downsample` function.

5.2 Non-Linearity Test

A bispectrum estimate —based purely on the dimensional fft, not a normalized bispectrum— was generated to test for non-linearities in the output y(t). The objective was to determine the presence of any non-linearities, not to fully characterize them. The bispectrum of y(t) is given in Figure 5. The bispectrum is non-zero and is not constant; however, the frequency distribution appears concentrated in the low frequencies (The bispectrum approaches zero asymptotically). Thus, the aerodynamics appear slightly non-linear.

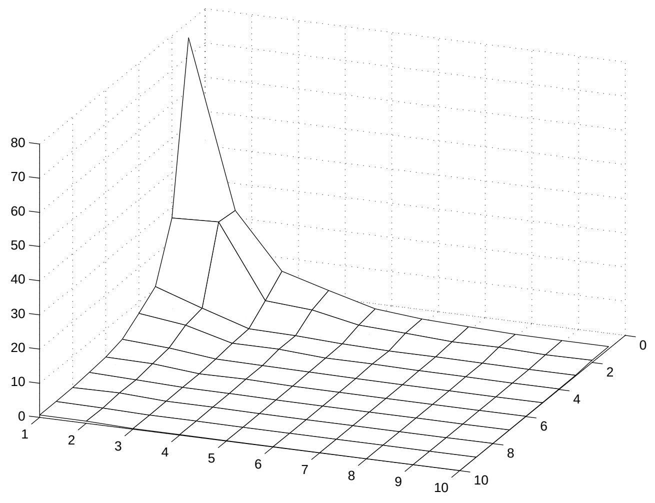

Figure 5: Bispectrum of y(t)

5.3 GPAC Order Estimation

Generalized partial autocorrelation (GPAC) techniques were used to estimate preliminary model orders. The G and H GPAC plots are given in Figure 6. The G-GPAC gives mixed results. A promising model order of nb=1, nf=1 (1,1) appears strong, but intuition suggests that the (1,1) model order only captures the quasi steady-state response. A second model order partially appears at (7,7) when excepting the rows after 10. Previous experiments with Box-Jenkins generated data often gives similar partial columns.

The H-GPAC gives two strong candidates: (0,1) and (1,2). The v(t) noise term's characteristics are unknown for this aerodynamic identification, but v(t) will need to account for the sampling rate truncation, CFD solution error, non-linear unsteadiness, etc. Intuition suggests that H(q) will need more than a (1,2) noise model.

6 Parameter Estimation and Diagnostic Testing

Parameter estimation is performed with a Levenberg Marquardt algorithm given in `lmpar.m` using the `estimate.m` Matlab script. Diagnostic testing is performed with the `estimate.m`, `gpac.m`, and `speccompare.m` scripts.

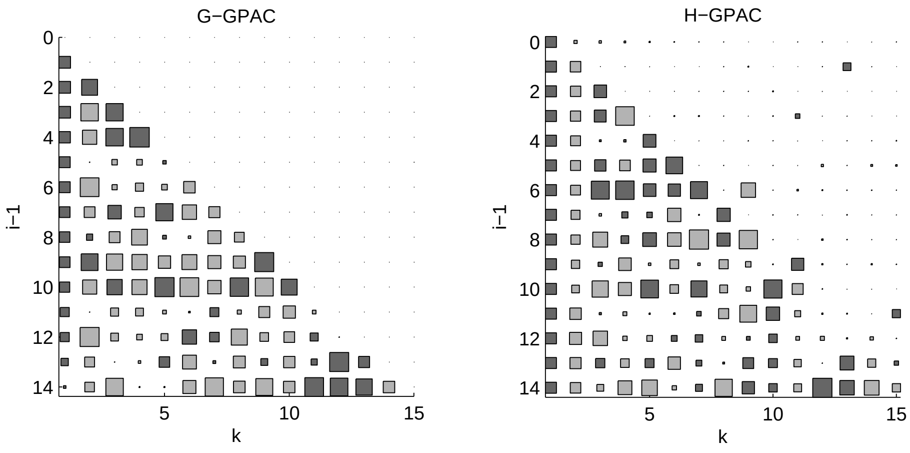

Figure 6: GPAC Order Estimate

6.1 Model 1: Quasi Steady State G(q) Model

The first parameter estimation is for the (1,1) model with the (1,2) noise model, which should only model the quasi steady state aerodynamics. The estimation gives low prediction errors ($y - \hat{y} \approx 10^{-6}$), but the transfer function spectrum is incorrect as shown in Figure 7. Despite the good training data fit, even the low frequency response is incorrect. Evidently, prediction error is a extremely poor model quality indicator! The (1,1) model is rejected.

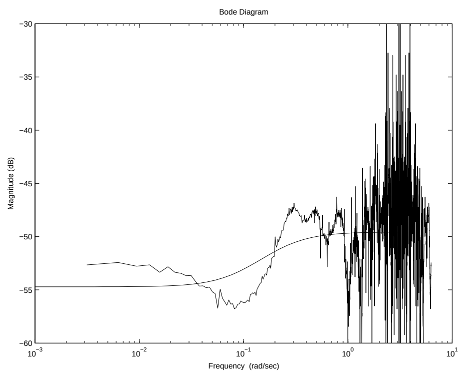

Figure 7: (1,1) Spectrum

6.2 Model 2: (7,7) Model

The next promising G(q) model starts with the nb=7, nf=7 (7,7) model with the (1,2) noise model. An initial parameter estimation gives excellent data fit.

However, the Q statistic for K=20 is 220, which is rather large. Significantly increasing the noise model order to (12,12) reduced Q to 14 with K=40, which is within the χ^2 tail limit.

Next, the G(q) and H(q) eigenvalues are determined for possible pole-zero cancellations. Iteration of parameter estimates and coincident eigenvalues suggests a model order of (4,4,7,7). The final eigenvalue plot is shown in Figure 8. Unfortunately, reducing the H(q) model order pushes Q=170 well outside the χ^2 tail limits. From this analysis, the plant model appears too low, which results in an overly complex noise model.

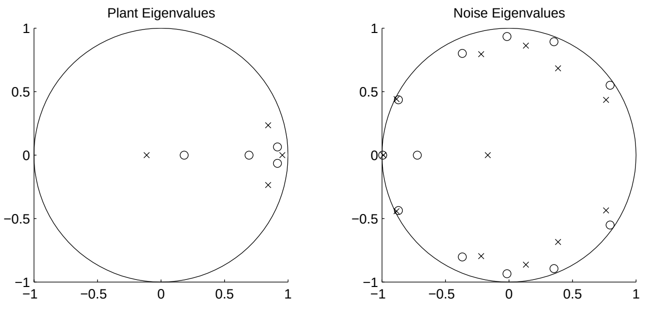

Figure 8: Model 2: (4,4,7,7) Eigenvalues

Model 2 will be rejected. For later comparison, the transfer functions G(q) and H(q) as modified with confidence intervals are shown below.

$$G(q) = \frac{0.0034 - 0.0094q^{-1} + 0.0091q^{-2} - 0.0036q^{-3} + 0.00046q^{-4}}{1 - 2.5q^{-1} + 2.1q^{-2} - 0.52q^{-3} - 0.063q^{-4}}$$

$$H(q) = \frac{1 - 1.2q^{-1} + 1.1q^{-2} - 0.47q^{-4} + 0.77q^{-5} - 0.28q^{-6} + 0.31q^{-7}}{1 - 2.3q^{-1}2.9q^{-2} - 1.8q^{-3} + 0.099q^{-4} + 0.93q^{-5} - 0.77q^{-6} + 0.28q^{-7}}$$

The bode plot is shown in Figure 9. Notice that the low frequencies are properly modeled, but that the higher frequency spectrum is too smooth. The high frequency spectrum differs from the Theodorsen theory; however the low frequency response is remarkably similar to theory!

6.3 Model 3: Higher Order G(q) Model

The previous results suggest using a higher order G(q) model to reduce the dynamics captured in the H(q) noise term. The order estimation section provided no general guidelines on this high of a model order. After some experimentation, a (20,20,5,5) model gave good results with a nice spectrum fit at the higher frequencies. This high order seems to indicate resampling is needed.

6.4 Model 4: Resampled

The sampling rate was increased to 1:20. This sampling rate is the maximum allowable to keep the interesting high frequency characteristics.

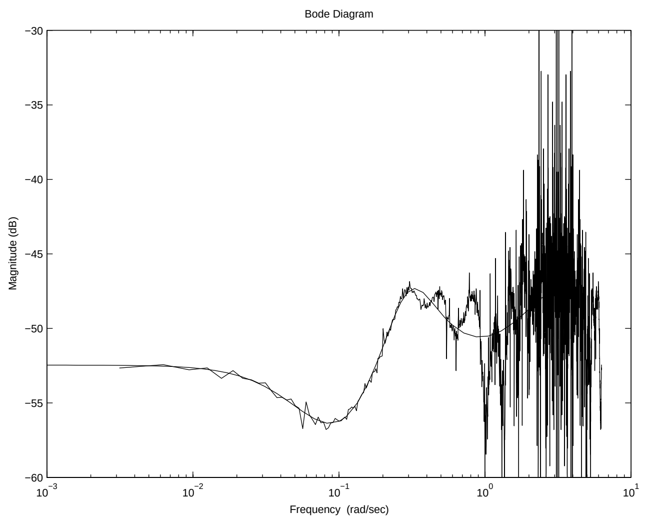

Figure 9: Model 2: Spectrum

Preliminary model order estimation with the GPAC seems to indicate possible models in the nb=nf=10 to 17 area (Fig 10) with H(q) orders of (1,1). Of course, these high orders are troublesome to find *exact* model orders. The high model orders seem to indicate inherent non-linear behavior.

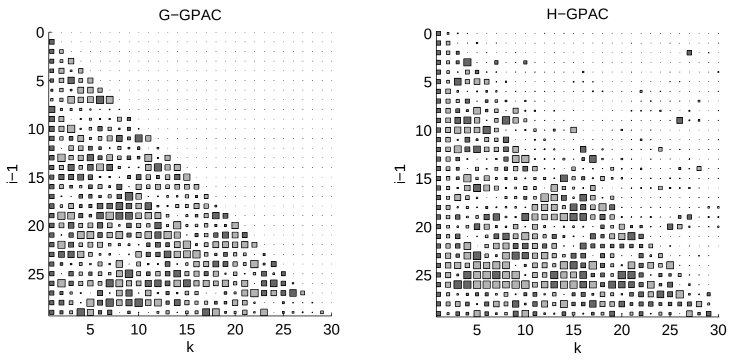

Figure 10: Model 4: GPAC

After some experimentation, a (10,10,4,4) model was selected. The Q and S statistics are still too large for K=30, but increasing the model order of either H(q) or G(q) creates pole-zero cancellations. The eigenvalues are shown in Figure 11. Notice that the aerodynamics seem to almost be a distributed system with an infinite number of eigenvalues spread around the unit circle. The G and H transfer functions are now quite long —21 terms for G!

The residual GPAC is shown in Figure 12. No improvements appeared feasi-

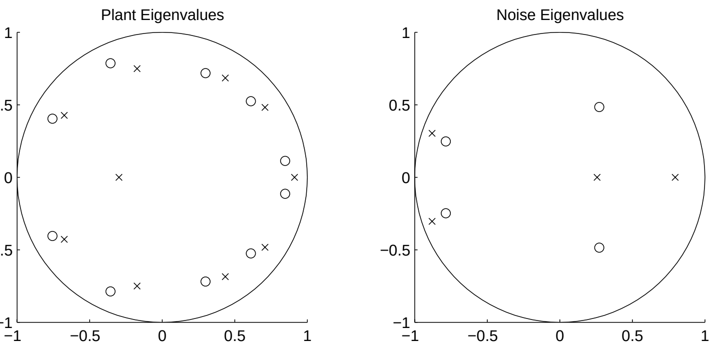

Figure 11: Model 4: Eigenvalues

ble based on this GPAC. Unfortunately, the input signal u(t) is filtered through a 2nd order system, so the equivalent white noise $\alpha(t)$ was created based on the structural input forcing function. Non-white inputs are troublesome for the diagnostic testing.

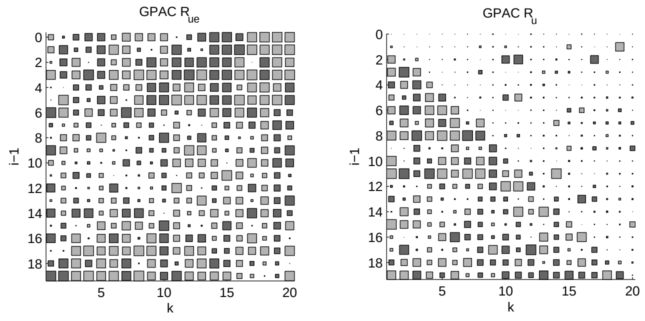

Figure 12: Model 4: GPAC

The final spectrum is shown in Figure 13. This model fits the data's spectrum even up into the higher frequencies.

7 Chosen Model

The final model #4 was chosen in spite of the model's complexity. The final transfer function has orders of: nb=10, nf=10, nc=4, and nd=4. The relatively large model order is needed to fit to the higher frequencies.

A final test involves validating the model against new system data. The CFD solver generated forces based on a chirp input signal. The CFD output forces (lines) are plotted against the Box-Jenkins model (dots) in Figure 14. The model appears to give a good response for significantly different input data (smooth chirp vs. white noise). Thus, the good prediction test is satisfied.

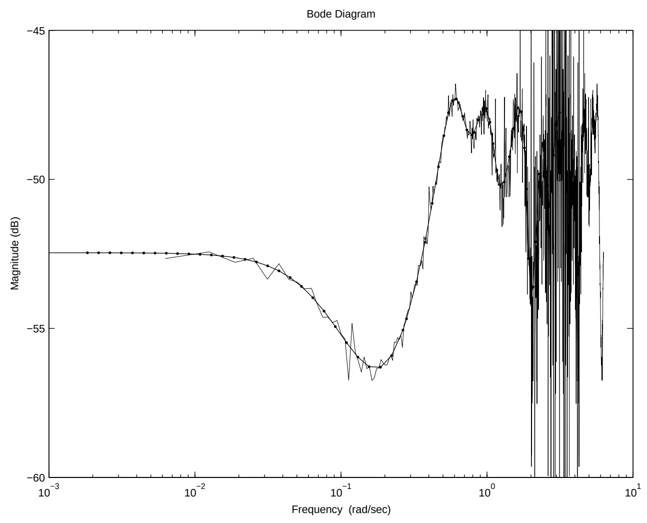

Figure 13: Model 4: Spectrum

8 Conclusions

This project investigated a Box-Jenkins model for aerodynamic system identification. The identification involved lift force estimation based on rotational angle input motions. An incompressible, inviscid, harmonic Theodorsen theory was used to develop an conceptual aerodynamic transfer function.

A Box-Jenkins model successfully modeled the low and moderate airfoil reduced frequencies, but was troublesome at higher frequencies. Interestingly, the Theodorsen theory qualitatively matches the CFD results except above approximately a reduced frequency of unity. The moderate frequency lift deficiency predicted by Theodorsen was found in the CFD solution and the Box-Jenkins model. Interestingly, as the high frequencies were resolved, the system appears more distributed. The final Box-Jenkins model accurately predicted the forces resulting from a validation input signal.

8.1 Difficulties

1. Differing Time Scales were caused by the difference between the CFD sampling rate and the Box-Jenkins sampling rate. This project's solution was to blindly sample the CFD output by truncating data points. A more sophisticated method would reduce the noise generated by truncation.

2. A non-white input resulted from a CFD requirement of consistent boundary conditions. Thus a white noise was input into a mass integrated (2nd order response) system. Two integrations created significant, but slow, variance in the actual displacement signal. This caused skewed inputs

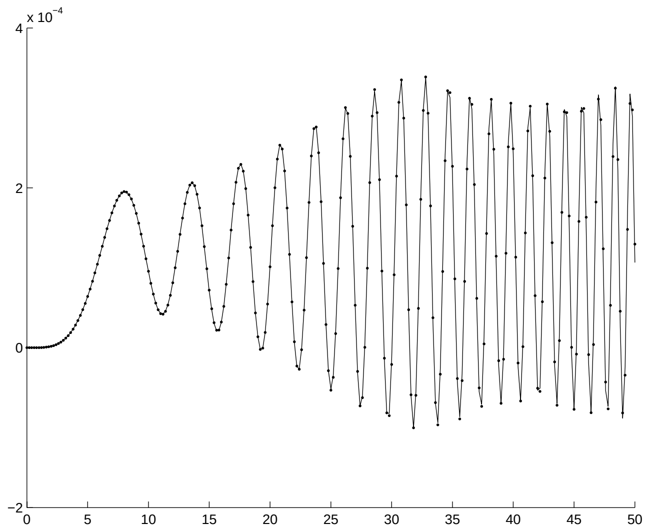

Figure 14: Chirp Validate

and outputs.

A better method might be to train with a restoring spring force and then remove the spring's contribution manually after the system identification.

3. The dominant aerodynamics appear linear, however non-linear properties were found with the bispectrum. In practice, the system model was requiring a significant number of eigenvalues.

4. Data set sizes were restricted by the CFD solver's speed. The high frequencies are overly noisy because of the small number of data points remaining after resampling.

References

Hagan, M., *ECEN/MAE 6423 System Identification Notes*, Fall 2004, OSU.
Mathworks, *Matlab Help*, 2002.
Hodges, D. and Pierce, G., *Introduction to Structural Dynamics and Aeroelasticity*, Cambridge Press, Cambridge, UK, 2002.

Nomenclature

L	Lift
U	Fluid Velocity
ρ	Fluid Density
S	Surface Area
c	Chord
b	Half Chord
a	Pitch Axis Offset
h	Heaving Displacement
α	Pitch Angle (Angle of Attack)
C_{L_α}	Coefficient of Lift per radian
k	Reduced Frequency, $\frac{\omega c}{2U}$

Code Listings

theo.m

```
%
%
%
clear all;

c=1;
U=500;
rho=0.002;
CLa=2*pi;
q=0.5*rho*U^2;
S= 0.02;

omega=logspace(-0, 4, 100)

c_k=besselh(1,2,omega*c/2/U)./(besselh(1,2,omega*c/2/U)+ i*besselh(0,2,omega*c/2/
    U))


%plot(real(c_k),imag(c_k))

%G=q*CLa*c*c_k + i * q*CLa*(c^2/4*U)*(1+c_k).*omega;
G=c_k + i * (c/4/U)*(1+c_k).*omega;
G=G;
Gtf=20*log(abs(G))
semilogx(omega,Gtf)
xlabel('Frequency');
ylabel('Transfer Function Magnitude, dB');
```

generatefrc.m

```
%
%



N=20000;
var_u=1000;
u=randn(N,1)*var_u^0.5;
u=[0; u];

o=zeros(N+1,1);

t=0:N;
x=[t' o u]; save 2dstab.frc x -ascii
plot(t,u)
```

orderest.m

```
%
%
%
%

%-------------------------------------------------------
% Initialize
%-------------------------------------------------------
SAMPLE_RATE=20;

load xn.dat
xn=downsample(xn,SAMPLE_RATE);
isteps=(0:(size(xn,1)-1))';
t=xn(:,2);
u=xn(:,4);
y=xn(:,7)-xn(1,7);
output=[isteps t u y];
save 2dstab.train1 output -ascii -double

% Get Raw Forcing Input
load 2dstab.frc
X2dstab=downsample(X2dstab,SAMPLE_RATE);
u_white=X2dstab(:,3);

% Setup the process
N=size(u,1);

%-------------------------------------------------------
% Estimate the impulse Response g(k)
%-------------------------------------------------------

% R shifted by N -> R(N)= R(tau=0)
Ru=xcorr(u);
Ruy=xcorr(y,u); %xcorr = E[ A(t) B(t-m) ] backwards!
KMAX=100;
R=zeros(KMAX,KMAX);
for tau=1:KMAX
    for i=1:KMAX
        R(tau,i)=Ru(abs(tau-i)+N);
    end
end
g=R^-1* Ruy(N:KMAX+N-1);

%-------------------------------------------------------
% Compute G GPAC from Impulse Response
%-------------------------------------------------------
NB_MAX=30;
NF_MAX=30;
```

```
G_phi=ggpac(g,NB_MAX,NF_MAX);

%-------------------------------------------
% Compute v(t) from y(t) u(t) and g(k)
%-------------------------------------------
ye=filter(g,1,u);
ve=y-ye;

%-------------------------------------------
% Estimate the autocorrelation function of v(t)
%-------------------------------------------
% R shifted by N -> R(N)= R(tau=0)
Rv_e=xcorr(ve);

%-------------------------------------------
% Compute H GPAC from acorr of v(t)
%-------------------------------------------
NC_MAX=30;
ND_MAX=30;
H_phi=hgpac(Rv_e,NC_MAX, ND_MAX, N);


%-------------------------------------------
% Plot the results
%-------------------------------------------
subplot(1,2,1)
title('G-GPAC');
plotphi(G_phi,NB_MAX,NF_MAX)
axis tight
subplot(1,2,2)
title('H-GPAC');
plotphi(H_phi,NC_MAX,ND_MAX)
axis tight

break
clf
hold off
title('H-GPAC');
plotphi(H_phi,NC_MAX,ND_MAX)
axis tight
```

ggpac.m

```
%
%    Computes G-GPAC
%    CO
function phi=ggpac(g,NB_MAX,NF_MAX)
warning off MATLAB:divideByZero
```

```
% Step through nb terms
for i=1:NB_MAX
    % Create g_i matrix
    clear gm
    for j=1:NF_MAX
        for k=1:NF_MAX
            index=(i-1)+(j-1)-(k-1);
            if( index+1>0)
                gm(j,k)=g(index+1);
            else
                gm(j,k)=0;
            end
        end
    end

    % Create g_(i+1) vector
    clear gv
    for j=1:NF_MAX
        index=(i-1)+(j-1)+1;
        if( index>0)
            gv(j,1)=g(index+1);
        else
            gv(j,1)=0;
        end
    end

    % Determine Phi term for each nf set
    for k=1:NF_MAX
        % gm/gv
        phi(i,k)=det(horzcat(gm(1:k,1:k-1), gv(1:k,1)))/det(gm(1:k,1:k));
    end


end
warning on MATLAB:divideByZero
```

hgpac.m

```
function phi=hgpac(Ry_e, NC_MAX, ND_MAX, OFFSET)

% Step through nb terms
for i=1:NC_MAX
    % Create g_i matrix
    clear gm
    for j=1:ND_MAX
        for k=1:ND_MAX
            index=(i-1)+(j-1)-(k-1);
            gm(j,k)=Ry_e(index+OFFSET);
        end
    end

    % Create g_(i+1) vector
    clear gv
    for j=1:ND_MAX
        index=(i-1)+(j-1)+1;
        gv(j,1)=Ry_e(index+OFFSET);
    end

    % Determine Phi term for each nf set
    for k=1:ND_MAX
        % gm/gv
        phi(i,k)=det(horzcat(gm(1:k,1:k-1), gv(1:k,1)))/det(gm(1:k,1:k));
    end

end
```

plotphi.m

```
function plotphi(phi,NB_MAX,NF_MAX)

axis ij equal
hold on
for i=0:NB_MAX-1
    nb=i+1;
    for j=1:NF_MAX
        nf=j;
            SCALE=abs(atan(phi(nb,nf).*1.25))*0.25;

        if(phi(nb,nf)>0)
            COLOR=[0.4 0.4 0.4];
        else
            COLOR=[0.7 0.7 0.7];
        end
        fill( SCALE*[-1 -1 1 1]+nf,SCALE*[-1 1 1 -1]+(nb-1),COLOR)
    end
end

xlabel('k');
ylabel('i-1');
```

estimate.m

```
%
%
%

N=size(y,1);
```

```
% Model Parameters
nb=1 +10;
nf=10
nc=4;
nd=4;
orders=[nb nf nc nd];

% Estimate Parameters
EPS=1.0E-10;
[theta,J]=lmpar(y,u,EPS,orders);

% Output Estimated Parameters
B=theta(1          : nb)'
F=[1 theta(nb+1      : nb+nf)']
C=[1 theta(nb+nf+1 : nb+nf+nc)']
D=[1 theta(nb+nf+nc+1 : nb+nf+nc+nd)']
[yhat, e_est]=boxres(y,u,theta',orders);
error=norm(e_est)
if(0)
    load e
    y_model=boxsim(u,e,theta',orders);
    actual_error=norm(y-y_model)
    plot(y_model,'r.-');
end
hold on;
%plot(y);
%plot(yhat,'.');
plot(yhat-y);

% Covariance Estimate
cov_parameters=norm(e_est.^2)/N*(J'*J)^-1;
parameter_dev=diag(cov_parameters).^0.5

% Estimate Q
e_est_cor=xcorr(e_est,e_est);
u_cor=xcorr(u_white,u_white);
K=30
Q=sum((e_est_cor(N+1:N+K)/e_est_cor(N)).^2)*N

% Estimate S
ae_cor=xcorr(e_est,u_white);
S=sum(ae_cor(N:N+K).^2/e_est_cor(N)/u_cor(N))*N

% GPAC Diagnostic
%[G_GPAC, H_GPAC]=gpac(u,e_est);
```

lmpar.m

```
%
%
%
%
function [finaltheta, finalJ]=lmpar(y,u,EPS,orders)

% CONSTANTS
MAX_MU=1.0E100;
DELTA=1.0E-6;
OUTER_ITERATIONS=300;
INNER_ITERATIONS=8;

ORDER=sum(orders);
nb=orders(1);
nf=orders(2);
nc=orders(3);
nd=orders(4);

N=size(y,1);

% Initilize Theta
theta=zeros(ORDER,1);

% Check Roots
F_est=theta(nb+1      : nb+nf);
D_est=theta(nb+nf+nc+1 : nb+nf+nc+nd);

mu=1;
[yhat, e_est]=boxres(y,u,theta',orders);
F_error=norm(e_est);

%--------------------------------
% Outer Loop
%--------------------------------
for k=1:OUTER_ITERATIONS
    fprintf('\n.');

    % Calculate Epsilon
    [yhat, e_est]=boxres(y,u,theta',orders);
    mu=mu/10;

    % Calculate J
    for i=1:sum(orders)
        theta_j=zeros(ORDER,1);
        theta_j(i)=DELTA;

        [yhat_j, e_est_j]=boxres(y,u,(theta+theta_j)',orders);
        [yhat_j2, e_est_j]=boxres(y,u,(theta-theta_j)',orders);
        J(:,i)=(yhat_j-yhat_j2) / (2*DELTA);
    end
    %--------------------------------
    % Inner Loop
    %--------------------------------
    for iter=1:INNER_ITERATIONS
        % Calculate Delta Theta
        delta_theta = ( J' * J + mu*eye(ORDER)) \ (J' * e_est);

        % Update Parameters
        theta_kk = theta + delta_theta;

        % Check Residuals
        [yhat_kk, e_est_kk]=boxres(y,u,theta_kk',orders);
        if(norm(e_est_kk) < F_error)
            success=1;
            fprintf('g');
            F_error=norm(e_est_kk);
            theta=theta_kk;
            break;
        else
            success=0;
            fprintf('b');
            mu=mu*10;
        end

    end

    % Stopping Criteria
    if((norm(2*J'*e_est_kk)*N <EPS) & (success==0))
        fprintf('\n Small Gradient Stopping Criteria at %d',k);
        break;
    end
    if((mu<EPS || mu>MAX_MU) & (success==0))
        fprintf('\n Mu Stopping Criteria at %d',k);
        break;
    end
end

%Conclusion
finaltheta=theta;
finalJ=J;
```

gpac.m

```
%
%
%
%
```

```
%------------------------------------------
% Initialize
%------------------------------------------
function [G_phi,H_phi]=gpac(u,y)

% Setup the process
N=size(u,1);

%------------------------------------------
% Estimate the impulse Response g(k)
%------------------------------------------

% R shifted by N -> R(N)= R(tau=0)
Ru=xcorr(u);
Ruy=xcorr(y,u); %xcorr = E[ A(t) B(t-m) ] backwards!
KMAX=80;
R=zeros(KMAX,KMAX);
for tau=1:KMAX
    for i=1:KMAX
        R(tau,i)=Ru(abs(tau-i)+N);
    end
end
g=R^-1* Ruy(N:KMAX+N-1);

%------------------------------------------
% Compute G GPAC from Impulse Response
%------------------------------------------
NB_MAX=20;
NF_MAX=20;
G_phi=ggpac(g,NB_MAX,NF_MAX);

%------------------------------------------
% Compute v(t) from y(t) u(t) and g(k)
%------------------------------------------
ye=filter(g,1,u);
ve=y-ye;

%------------------------------------------
% Estimate the autocorrelation function of v(t)
%------------------------------------------
% R shifted by N -> R(N)= R(tau=0)
Rv_e=xcorr(ve);

%------------------------------------------
% Compute H GPAC from acorr of v(t)
%------------------------------------------
NC_MAX=20;
ND_MAX=20;
H_phi=hgpac(Rv_e,NC_MAX, ND_MAX, N);
```

```
%------------------------------------------
% Plot the results
%------------------------------------------
subplot(1,2,1)
title('GPAC_R_{ue}');
plotphi(G_phi,NB_MAX,NF_MAX)
axis tight

subplot(1,2,2)
title('GPAC_R_u');
plotphi(H_phi,NC_MAX,ND_MAX)
axis tight
```

speccompare.m

```
%
%
%

N=size(u,1);
n=0:N-1;

% Parameters
%A=[]';
%B=[]';

A=F
B=B

TIMEMAX=1;

% Window
if(0) % 3 term Blackman-Harris
     a0=0.42323;     a1=0.49755;    a2=0.07922;
     w=a0-a1*cos(2*pi*n/N)+a2*cos(2*pi*2*n/N);
elseif(1) % Hanning
     w=sin(pi*n/N).^2;
elseif(0) % Triangle Window
     w=1.0-abs(n-N/2)/(N/2);
else % rectangular
     w=ones(1,N);
end

yw=y.*w';
uw=u.*w';

% FFT
Y=fft(yw);
U=fft(uw);
```

```
S= (conj(Y).*Y) ./ (conj(U).*U);

% Frequency Method
G=filt(B,A,1);

% Plotting
semilogx(n/N*2*pi, 10*log10(S),'r'); hold on
xlabel('Frequency (rad/s)');
ylabel('Transfer Function |G(e^{jw})|');
bodemag(G);hold on
axis([0.001 10 -60 -30]);
```