Computational Aids in Aeroservoelastic Analysis Using MATLAB™

Changho Nam • Youdan Kim • Terrence A. Weisshaar

Chapter 1
Introduction to MATLAB

1.1 General MATLAB Information

MATLAB is a high-performance language for technical computing, and is an interactive system whose basic data element is a matrix. These days, MATLAB is widely used for (i) mathematics and computation, (ii) algorithm development, (iii) modeling, simulation, and prototyping, data analysis, exploration, and visualization, (iv) scientific and engineering graphics, and application development including graphic user interface building.

These days, many specialized toolboxes have been developed: Control Systems, Signal Processing, Optimization, Neural Network, System Identification, and so on. Those toolboxes provide MATLAB very powerful tools for various and practical applications.

The purpose of this chapter is to introduce a number of MATLAB functions that are widely used for dealing with engineering problems. Because there exist a lot of MATLAB functions, only selected MATLAB functions will be provided in the following sections. Especially, some basic MATLAB functions used in this book will be briefly summarized.

1.2 Introduction to MATLAB

When you get into the MATLAB, MATLAB displays a prompt (>>) indicating that MATLAB is ready to receive a command. When you type *help* command, you can obtain on-screen information about MATLAB commands which are useful for users. And when you type *quit* at the MATLAB prompt, you can exit MATLAB anytime. A brief summary of some of MATLAB commands is provided in the following sections for readers who are not familiar with MATLAB. Especially, some MATALB commands for matrix operation and solution as well as plotting commands for visualization of the numerical results are explained.

1.2.a Vectors and Matrices

Row vectors are entered by typing a list of numbers in brackets as follows:

```
>> x = [ 0 3 2]
```

Note that elements are separated using either spaces or commas. To create column vectors, transpose the vector using the apostrophe (') key. When a complex vector is transposed, the *conjugate transpose* of the vector will be obtained. To obtain the unconjugated transpose, type a dot before the transpose symbol (.').

Matrices are entered row by row, and each row is separated by semicolon. For example, to enter 3-by-3 matrix, type the following command.

```
>>  A = [ 0 3 2; 5 1 2; 9 -6 7]
```

Note that a semicolon is used to indicate the end of each row.

The *workspace* is the area of memory used by the MATLAB. Two commands, *who* and *whos*, provide the information of current variables in the workspace. The command *who* lists all the variables in memory, while *whos* displays the variables, sizes and storage information. The command *size(A*) returns the number of rows and

columns of a matrix A, while the command *length(x)* returns the dimension of a vector x.

The element in row i and column j of a matrix A is denoted by A(i,j). For example, A(2,3) is the number in the second row and third column. The colon operator (:) represents all the rows or all the columns of a matrix, and therefore the colon operator can be used to define vectors form a matrix. Consider the following expressions.

```
>>  x=A(3, :)
 >>  y=A(:, 2)
```

By typing the above commands, x gives a row vector of third row of matrix A, and y gives a column vector of second column of matrix A, respectively. Also, A(i:j, k:l) returns rows i through j and columns k through l of matrix A, and therefore the following expression yields a two-by-two submatrix of A.

```
>>  A(1:2, 1:2)
```

When output variable is not specified, MATLAB uses the variable *ans* to store the results of a calculation, and *ans* keeps the variable value of most recent answer

Rows and columns can be deleted from a matrix using just a pair of square brackets. To delete the first column of matrix A, type the following command.

```
>>  A(:,1) = [ ]
```

Concatenation is the process of joining small matrices to make bigger ones. For the appropriately dimensioned matrices A, B, C, and D, the following command can be used to define a new matrix.

```
>>  E = [A B; C D]
```

The command *clear* clears specified variables from the workspace. For example, to clear variables x and y from the workspace, type the following command.

```
>> clear x y
```

Typing *clear* by itself clears the entire workspace.

1.2.b Basic Scalar and Matrix Computations

Conventional decimal notation is used in MATLAB, and scientific notation an also be used by using the letter e to specify a power-of-ten scale factor. Several predefined functions provide special and useful constant values as follows:

pi	The value of π, (= 3.141592 …)
i or j	Imaginary unit, (= $\sqrt{-1}$)
eps	Floating-point relative precision for the computer being used
Inf	Infinity
NaN	Not-a-Number

MATLAB provides several functions that generate special matrices. The *zeros(m,n)* function generates an m-by-n matrix containing all zeros, the *ones(m,n)* function generates an m-by-n matrix containing all ones, the *eye(n)* function generates n-by-n identity matrix, respectively. Also, *rand(m,n)* function generates an m-by-n matrix with uniformly distributed random elements.

MATLAB supports basic algebraic operations: addition (+), subtraction (−), multiplication ( ), division (/), left division (\), and exponentiation or power (^). MATLAB also supports the elementary math functions including standard trigonometric functions, hyperbolic functions, and computing the absolute value and the square root. For example,

```
>> a1 = sin(x)
>> a2 = cosh(x)
>> a3 = asin(x)
>> a4 = abs(x)
>> a5 = sqrt(x)
```

```
>>  a6 = exp(x)
>>  a7 = log(x)
```

The above commands perform the well-known math functions, and yield the sine value, the cosine hyperbolic value, the arcsine or inverse sine value, the absolute value, the square root value, the exponential value, and the natural logarithm value of the variable x, respectively.

Complex variable x can be defined by the following command.

```
>>  x = 1 + 2    i
```

MATLAB supports several functions for treating complex numbers.

real(x)	This function computes the real part of x
imag(x)	This function computes the imaginary part of x
abs(x)	This function computes the magnitude of x
angle(x)	This function computes the phase angle of x
conj(x)	This function computes the complex conjugate of x

Matrix addition, subtraction, and multiplication are performed according to linear algebra rules. For example,

```
>> C = A + B
>> C = A − B
>> C = A    B
```

The above operations define a new matrix C as sum of two matrices, subtraction matrix B from matrix A, and matrix multiplication, respectively.

Other useful matrix commands are listed as follows with a brief description:

sum(A)	This function performs the sum over each column
A'	This function performs the conjugate transpose of matrix A

diag(A) This function picks off the diagonal of A
det(A) This function produces the determinant of matrix A
trace(A) This function produces the sum of diagonal elements of matrix A
rank(A) This function produces the rank of matrix

Matrix inversion is defined by

```
>> A\B
>> A/B
>> inv(A)
```

In the above expressions, A\B is equivalent to *inv(A) B*, and A/B is equivalent to *B inv(A)*, respectively.

Other useful matrix functions such as *eig* (eigenvalue), *cond* (condition number), *expm* (matrix exponential), and *norm* are discussed in detail in the subsequent section.

1.2.c Format Commands and File Manipulation

All computations in MATLAB are preformed in double precision. However, the display format of data can be modified using the *format* command. For example, the displayed output can be controlled using the following commands.

```
>> format short
>> format long
>> format short e
>> format long e
```

The command *format long* displays the output in 15 digit scaled fixed point, while the command *format long e* displays in scientific notation (15 digit floating point). Note that the chosen format remains in effect until changed.

All variables in MATLAB workspace can be saved in the binary MATLAB format by using the following *save* command.

```
>> save filename
```

If no extension is specified, MATALB adds the extension .MAT to the filename. Only selected variables can be saved by performing a partial save.

```
>> save filename x y
```

where variables x and y are saved in a file name filename.mat. Variables can be saved in a standard text format.

```
>> save data.dat x y /ascii /double
```

The above command save variables x and y in a file name data.dat in 16-digit ASCII format. Without *double* option data is saved in 8-dgit ASCII format.

To restore the variables in the file that was generated by *save* command, use the *load* command.

```
>> load filename
```

Only specified variables can be loaded by using the following command.

```
>> load filename x y
```

The command *what* shows the MATLAB files and the MAT data files under the current directory, and the command *dir* shows all the files under the current directory. The command *delete* deletes files, the command *cd* changes directory, and the command *type* lists the contents of the file, respectively.

The command *diary* is used to create all the input commands and displayed outputs in a MATLAB session. To create the MATLAB session in a file with a different name, use

```
>> diary filename
```

To close diary file, use

```
>> diary off
```

Now, all the command and answer lines that are displayed on the screen are saved under a given name in ASCII format, and the file can be edited later.

1.2.d Matrix Functions

MATLAB supports a number of matrix functions that manipulates the contents of a matrix, which is one of the features of the MATLAB.

Basic Matrix Functions
Matrix exponential, logarithm, and square root can be obtained by the following commands.

```
>> expm(A)
>> logm(A)
>> sqrtm(A)
```

Note that exp(A) (that is without the m) gives exponential of matrix A element-by-element.

Norms
The norm of a matrix is a scalar that gives some measure of the size of the matrix. Several different definitions are commonly used.

```
norm(A)          Largest singular value of A, max(svd(A))
norm(A,2)    Same as norm(A)
norm(A,1)    1-norm of A, the largest column sum, max(sum(abs((A))))
norm(A,inf) Infinity norm of A, the largest row sum, max(sum(abs((A')))).
norm(A,'fro')          Frobenius norm, sqrt(sum(diag(A'*A))).
```

Note that the command *norm* is also used to compute the norm of vector. For a vector x, the command *norm(x)* produces the Euclidean norm of vector x.

Condition Number

The command *cond* returns the condition number (the ratio of the largest singular value of matrix A to the smallest of matrix).

```
>> cond(A)
```

The condition number represents a quantitative measure of ill-conditioning(singlularity) of a matrix. Large condition numbers indicate a nearly singular matrix, and therefore the condition number of a singular matrix is infinity. If you try to compute the inverse of singular or nearly singular matrix, then you will get a warning message telling you that the computed results may be inaccurate.

Solution of Linear Equations

Consider a set of simultaneous linear algebraic equations in matrix notation.

$$y = A x$$

where xë R^n and yë R^m. The above problem has three possibilities: (i) exactly one solution, (ii) an infinity of exact solutions, and (iii) no exact solution. For a determined case [m=n=rank(A)], the solution is obtained by the following command.

```
>> x = inv(A)   y
```

or

```
>> x = A \ y
```

The second and third cases are under-determined case and over-determined case, respectively, and the following command yields the minimum norm solution and least square solution, respectively.

```
>> x = pinv(A)   y
```

The Eigenvalue Problem

Consider an eigenvalue problem of a matrix

$$A\,x = \lambda\,x$$

where λ is the eigenvalue of matrix A, and x is the associated eigenvector. The eigenvalue problem can be solved using the *eig* command.

```
>> lambda = eig(A)
>> [V, D] = eig(A)
```

The command *eig* produces a vector lambda including the eigenvalues of matrix A, a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors, respectively.

For the square matrices A and B, the generalized eigenvalue problem is defined as follows.

$$A\,x = \lambda\,B\,x$$

Then, the following command returns a diagonal matrix D of generalized eigenvalues and a square matrix X whose columns are corresponding eigenvectors.

```
>> [V, D] = eig(A, B)
```

Singluar Value Decomposition

Any rectangular matrix can be decomposed as follows.

$$A = U\,\Sigma\,V^{H}$$

where supscript H denotes the Hermitian or conjugate transpose of matrix, matrices U and V are unitary matrices, and the matrix Σ is a diagonal matrix containing as its

elements the real (non-negative) singular value of matrix A. The command *svd* is used for singular value decomposition of a matrix.

```
>>  [U, S, V] = svd(A)
>>  sigma = svd(A)
```

Note that the diagonal matrix S and the vector sigma include the singular values in decreasing order.

MATLAB supports various matrix decomposition methods such as QR decomposition (command *qr*), Cholesky decomposition (command *chol*), LU decomposition (command *lu*), and so on, which will not be discussed in this section. More information can be found in MATLAB manual or using the on-line help commands.

1.2.e Data analysis

MATLAB uses column-oriented analysis for multivariable statistical data, and MATLAB has data analysis functions for a vector or a column of a matrix.

The following commands can be used to determine maximum and minimum, and their locations.

```
>>  y = max(x)
>>  y = min(x)
>>  [y, k] = max(x)
>>  [y, k] = min(x)
```

If an input variable is a vector, then the function returns the largest or smallest value of the input vector. However, if an input variable is a matrix, then the function returns a row vector containing the maximum or minimum element from each column. And k is an index of the maximum value in the vector.

To sort a vector in the ascending order, the command *sort* can be used.

```
>>  [y, i] = sort(x)
```

The sorted values are returned to the vector y and the reordered indices are stored in the vector i. When x is complex, the elements are sorted by the magnitude of x.

The mean value and the standard deviation of a vector can be calculated by the following commands.

```
>>  mean(x)
>>  std(x)
```

1.2.f Polynomials

In MATLAB, polynomials are represented as vectors containing the polynomial coefficients in descending order. For example, $x^3 + 3x^2 + 3x + 1$ can be represented as follows:

```
>> y = [1 3 3 1]
```

The roots of the polynomial equation $x^3 + 3x^2 + 3x + 1 = 0$ are solved using the command *roots*.

```
>> y = [1 3 3 1]; r=roots(y)
```

The roots of the polynomial equation may be reassembled back into the original polynomial equation with the command *poly*.

```
>>  poly(r)
```

The command poly returns the characteristic equation of matrix, when the input variable is a matrix.

```
>>  poly(A)
```

A polynomial can be evaluated at a point using the command *polyval*, and the command *polyval* evaluates polynomials in the matrix sense.

```
>> polyval(y, val)
>> polyvalm(y, A)
```

where y is a polynomial and val is the point at which the polynomial is to be evaluated.

The product of the polynomials, the convolution of the coefficients, can be performed by the command conv, while a polynomial can be decomposed by the deconvolution command.

```
>> conv(x, y)
>> [m, r] = deconv(a, b)
```

where m is the quotient and r is the remainder, that is $a(s) = b(s)m(s) + r(s)$.

The partial fraction expansion can be performed by the command *residue*.

```
>> [r, p, k] = residue(num, den)
```

The above command finds the residues, poles and direct term of a partial fraction expansion of the ratio of two polynomials num(s)/den(s). If there are no multiple roots, the residues are returned in the column vector r, the pole locations in column vector p, and the direct terms in row vector k. The direct term coefficient vector is empty if the function num(s)/den(s) is strictly proper, or length(num) < length(den). The residue command can also be used to form the (num, den) polynomials from its partial fraction expansion:

```
>> [num, den] = residue(r, p, k)
```

To generate a polynomial curve that fits a given set of data, the command *polyfit* can be used.

```
>>  p = polyfit(x, y, n)
```

where x and y are vectors of the given data set in (x, y) form. The above command finds the coefficients of a polynomial p of degree n that fits the data in a least squares sense, that is the polynomial is obtained by minimizing the error between the polynomial and the given data set.

1.2.g Plotting and Graphics

MATLAB provides a powerful graphic tools that display vectors and matrices in a variety of formats: x-y plots, polar plots, bar graphs, contour plots, and 3-D plots. In this section, only a few of graphics functions will be described.

The command *plot* plots the vector y versus the vector x.

```
>>  plot(x,y)
```

Note that the vectors must have the same length. To display the graphs in a logarithmic scale, following commands are used.

semilogx(x, y)	x-axis (logarithmic), y-axis (linear)
semilogy(x, y)	x-axis (linear), y-axis (logarithmic)
loglog(x, y)	both axes (logarithmic)

To open a new figure window, the command *figure* is used. The command *plot* erases the previous plots and reset all axis properties before drawing new plots. The command *hold on* holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph, and the command *hold off* returns to the default mode. The sample plotting command is shown below.

```
>>  t = 0 : pi/100 : 2  pi;
```

```
>> y = sin(t);
>> plot(t, y)
>>
>> figure
>> y2 = sin(t - 0.25);
>> plot(t, y2)
>> hold on
>> plot(t, y+y2)
```

In the above example, t=0:pi/100:2 pi represents a vector t which starts from 0 and ends at (2 pi) with an interval of (pi/100).

Multiple curves can be plotted against the same axis.

```
>> plot(x1, y1, x2, y2, x3, y3)
```

The variables (x1, y1), (x2, y2), and so on, are pairs of vectors, and therefore the above command plots y1 versus x1, y2 versus x2, and y3 versus x3. In this case, x1, x2, and x3 can have different length. However, xi and yi must have the same length (i.e., same number of data points).

The command *plot* generates a line plot that connects the points with line segments. Different line type can be used by line option, and also a point plot can be plotted instead of line plot. For example, the following command plots a red dashed line with a x-mark at each data point;

```
>> plot(x,y,'rx--:')
```

And the following command plots green circle at each data point but does not draw any line.

```
>> plot(x,y,'go')
```

Various line types, plot symbols and colors may be obtained with the following command

```
>> plot(x1, y1, 's1', x2, y2, 's2', ...)
```

where s1 and s2 are characters string made from one element from any or all the following options.

Colors:	indicator	color	indicator	color
	r	red	g	green
	b	blue	w	white
	y	yellow	k	black
	m	magenta	c	cyan
Lines:	indicator	line type	indicator	line type
	-	solid	:	dotted
	-.	dashdot	--	dashed
Points:	indicator	point type	indicator	point type
	.	point	o	circle
	x	x-mark	+	plus
	v	triangle (down)	d	diamond
	^	triangle (up)	s	square
	<	triangle (left)	*	star

After plotting a graph on the screen, grid lines, title, x-axis and y-axis label may be inserted. The following commands are used for the plot output.

```
>> grid on
>> title('Title of graph')
>> xlabel('Time (sec)')
>> ylabel('ydata')
```

The command *grid off* takes grid off, and the command *grid*, by itself, toggles the grid state. Also, a text can be written at the specified point on the graphic screen using the following command.

```
>>  text(x,y,‘text contents’)
```

where (x, y) are the coordinates using the axes from the current plot, and text is written beginning at a point (x,y).

```
>>  text(x,y,‘text contents’, ‘sc’)
```

The above command inserts the text on the graphic screen at the point (x,y) assuming that the lower-left corner is (0,0) and the upper-right corner is (1,1).

The command *gtext* writes the text at the position on the graphic screen indicated by the mouse or arrow keys. That is,

```
>>  gtext(‘text contents’)
```

MATLAB automatically scales the axes to fit the data values. The following *axis* command sets the axes to the specified setting.

```
>>  axis( [ xmin xmax ymin ymax ] )
```

The command *axis*, by itself, freezes the current setting for subsequent plots, and a second execution of the *axis* command resumes auto-scaling. There are more options in the *axis* command as follows.

axis squre	Makes x-axes and y-axes the same length
axis equal	Makes the individual tick mark increments on the x-axes and y-axes the same length
axis auto	Returns the axis scaling to its default
axis on	Turns on axis labeling and tick marks
axis off	Turns off axis labeling and tick marks

The above commands are useful when comparing curves from different plots.

The command *subplot* is used to split the graph window into subwindows, and therefore multiple plots can be displayed in the same window.

```
>>  subplot(m, n, p)
```

or

```
>>  subplot(mnp)
```

The above command breaks the figure window into an m-by-n matrix of small subplots and selects the p-th axes for the current plot. The subplots are numbered along first the top row of the figure window, then the second row, and so on. The sample plotting command for plotting data in four different subwindows is shown below.

```
    >>  t = 0 : pi/100 : 2  pi;
>>  y1 = sin(t);
>>  y2 = cos(t);
>>  y3 = sinh(t);
>>  y4 = cosh(t);
>>  subplot(2, 2, 1); plot(t, y1); title('sin(x)')
>>  subplot(2, 2, 2); plot(t, y2); title('cos(x)')
>>  subplot(2, 2, 3); plot(t, y3); title('sinh(x)')
>>  subplot(2, 2, 4); plot(t, y4); title('cosh(x)')
```

There are several commends for screen control. The commend *pause* is useful when we display multiple graphic windows subsequently. The *pause* command causes a procedure to stop and wait for the user to strike any key before continuing. And the command *pause(n)* pauses for n seconds before continuing. The commend *clf* clears the current figure window, and the command *cla* clears the current plot.

MATLAB supports extensive graphic facilities. In this section, only some commends for plotting 2-D (x-y) graphs are discussed. Contour plots, 3-D plots, mesh plots with different color scales can be displayed in MATLAB, and detailed descriptions can be found in the Help menu as well as the MATLAB user manual.

1.3 Programming in MATLAB

1.3.a M-Files

MATLAB can execute sequences of commands that are stored in files that are called M-files. There are two kinds of M-files: Script M-files and function M-files that cam be created by a text editor. Script M-files are useful for automating a series of commands you want to execute many times. Script M-files operates on data in the workspace, and do not accept input arguments nor return output arguments. Function M-files are useful for adding more subroutines to MATLAB for specific applications, and therefore accept input arguments and return output arguments. M-files can be called from the command window, or from within another M-file.

Consider the example of script M-files that is saved under the name of test1.m.

```
% finding roots for second order algebraic equation
a = 1; b = 3; c=2;
D=b  b – 4  a   c;
root(1) = (-b +sqrt(D))/(2  a);
root(2) = (-b -sqrt(D))/(2  a);
```

The symbol % is used to add comments to the program, and everything to its right on that line is ignored and is not executed. To perform the series of commands in the test1.m file, the command *test1* is used as follows

```
>>  test1
```

The above script M-file can be written as a function M-file under the name of test2.m as follows.

```
function  root = test2(a,b,c)
% finding roots for second order algebraic equation
D=b  b – 4  a   c;
root(1) = (-b +sqrt(D))/(2  a);
root(2) = (-b -sqrt(D))/(2  a);
```

The work of the script M-file test1.m can also be done by the function M-file test2.m as follows:

```
>>  a = 1; b = 3; c=2;
 >>  root = test2(a,b,c);
```

MATLAB has five flow control statements (if statements, switch statements, for loops, while loops, and break statements) that support some basic programming structures that allow looping and conditioning commands along with relational and logical operators. These features are briefly summarized in the subsequent sections.

1.3.b Relational and Logical Operators

MATLAB has six relational operators for comparing two scalrars (or matrices of equal size, element by element) as follows:

Relational Operator	Interpretation
<	less than
<=	less than or equal
>	greater than
> =	greater than or equal
==	equal
~=	not equal

For complex data, “==” and “~=” compare both the real and imaginary parts; other operators compare only the real part.

Two logical expressions can also be combined using the logical operators. There are three logical operators available in MATLAB; they are

Logical Operator	Symbol
and	&
or	|
not	~

1.3.c If Statements

The *if* statement evaluates a logical expression and executes a series of statements when the expression is true. The optional *elseif* and *else* statements provide for the execution of alternate groups of statements. The syntex is shown below:

```
If      logical expression #1
        statement group #1
elseif  logical expression #2
        statement group #2
else
        statement group #3
end
```

If logical expression #1 is true, the first statement group is executed, if logical expression #2 is true, then the second statement group is executed; otherwise, the third statement group is executed. Note that the *end* statement is mandatory. The *if* statement can also be nested; *if* statement can be included in the statement group.

Consider the following example.

```
>>  n = 3.14;
>>  if  n > 0
>>    disp('positive number')
>>  elseif  n < 0
>>    disp('negative number')
```

```
>> else
>>   disp('zero')
>> end
```

In the above example, the command *disp* displays the given string or variable.

Several functions are very useful with *if* statement: these functions are *any*, *all*, *isequal*, *isempty* and so on, which will not be discussed in this book. The interested readers may refer to the user manual or on-line help command.

1.3.d For Loops

The *for* statement is used to execute a series of statements a predetermined number of times. The syntax is shown below:

```
for     index = expression
        statement group
end
```

Note that the index of the *for* loop must be a variable, and the index contains the last value used after completing the *for* loop. There may be also multiple loops. The colon operator is used to define the index expression using the following format:

```
For  iter = ni : nd : nf
```

Where iter is a loop index which starts from ni and ends at nf with an increment of nd. If nd is not defined, the default value of nd=1 is used. The value of iter would be ni, then ni+nd, then ni+nd+nd, and so on until the final value of nf.

Consider the following example including multiple loops.

```
>> A=rand(3,4); B=rand(4,5);
>> C=zeros(3,5);
>> for  i = 1: 3
```

```
>>       for  j = 1:5
>>              for  k  = 1:4
>>                     C(i,j) = C(i,j) + A(i,k)    B(k,j)
>>              end
>>       end
>>  end
```

The *break* statement can be used to exit a loop before it has been completed. This statement is useful if an error condition is detected within the loop. Therefore, *break* statement is usually used with *if* statement, because *if* statement may be used to check the escape condition. In nested loops, *break* exits from the innermost loop only.

1.3.e While Loops

The *while* loop executes a group of statements repeatedly as long as the controlling expression is true. The syntax is shown below.

```
while    expression
         statement group
end
```

Consider the following example

```
   >>  n = 1;
>>  while n  <= 100
>>    n = n + 1;
>>  end
```

The *break* statement can also be used in the *while* loop to exit a loop before it has been completed.

1.3.f Switch Statement

The *switch* statement executes a group of statements based on the value of a variable or expression. The syntax is shown below.

```
switch switch_expression (scalar or string)
  case case_expression1
        statement group
  case {case_expression2, case_expression3}
        statement group
  otherwise
        statement group
end
```

The statement group following the first *case* where the switch_expression matches the case_expression is executed. When the case expression is a cell array (as in the second case above), the case_expression matches if any of the elements of the cell array match the switch expression. If none of the case expressions match the switch expression then the *otherwise* case is executed (if it exists). Only one *case* is executed and execution resumes with the statement after the *end*. Consider the following example.

```
   >> switch var
>>   case 0
       disp('0')
>>   case {-1, +1}
>>         disp('-1 or +1')
>>      otherwise
>>         disp('something else')
>> end
```

1.3.g Other Useful Commands

return

Normally functions return when the end of the function is reached. A *return* statement can be used to force an early return.

global

Each MATLAB function has its own local variables separated from those of other functions, and from those of the base workspace and non-function scripts. By declaring the variable as *global* in all the functions, several functions can share a single copy of that variable. Any assignment to that variable, in any function, is available to all the other functions declaring it *global*. Consider the following example.

```
>> function [x,y] = projectile(v, gamma, t)
  >> global  gravity
  >> x = v    t    cos(gamma);
  >> y = v    t    sin(gamma) – 0.5    gravity * t^2;
```

1.3.h Basic Input-Output Functions

input

The value of variables can be entered through the keyboard using the *input* command that displays a text then waits for input. The values entered by user are stored in the variable specified. Consider the following example.

```
  >> var1 = input(‘Enter values for var1’);
```

When the above command is executed, the text ‘Enter values for var1’ is displayed on the screen. Then the user can enter an expression that specifies values for var1. For character string input string input, use the following option.

```
  >> R = input('What is your name' , 's')
```

disp

The *disp* command displays a string of text or numerical values on the screen. Consider the following example.

```
>> disp(‘This is a MATLAB displacement command test.’)
```

fprintf

The *fprint* command displays a text and values on the screen with specified format. The syntax is shown below.

```
fprintf(format, variable)
```

where the format contains the text and format specifications. Within the text, %e and %f are used for exponential notation or fixed point notation, respectively. If the string \n appears in the format, the text statements before \n is displayed, and the rest of the information will be displayed on the next line. A simple example is shown below.

```
>> temp=18;
>> fprint('The temperature is  %f degrees C \n', temp)
```

The *fprintf* command can also write formatted data to file using the file open and close commands *fopen* and *fclose* which will not be discussed in this book. Again, the interested readers may refer to the user manual or on-line help command.

1.4 Numerical Analysis using MATLAB

MATLAB supports lots of numerical techniques that are useful for solving many engineering problems. MATLAB functions for solving ordinary differential equation, nonlinear algebraic equation, and numerical integration will be briefly summarized in this section.

1.4.a Ordinary Differential Equations

Linear and nonlinear differential equations can be solved using MATLAB. Higher-order differential equation can be written as a system of coupled first-order differential equations. A first-order ordinary differential equation (ODE) can be written in the following form.

$$\dot{x} = \frac{dx}{dt} = f(t, x)$$

To integrate the above ODE, the function representing the ODE should be defined as the function M-file, and the integrating function calls the function M-file. The most

popular methods for integrating a first-order differential equation are Runge-Kutta methods. The syntax of Runge-Kutta fourth and fifth order algorithm is shown below.

```
>> [t, x] = ode23('funct', t0, tf, x0)
```

where 'funct' is a function containing the derivative information, t0 and tf are the initial and final time, respectively, and x0 is an initial conditions. For example, let us consider the van der Pol equation.

$$\ddot{x} - 3(1 - x^2)\dot{x} + x = 0$$

The above equation can be written as

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = 3(1 - x_1^2)x_2 - x_1$$

where $x_1 = x$ and $x_2 = \dot{x}$. To integrate the above equations, function M-file *vandP.m* should be first defined as follows.

```
function dy = vandP(t,x)
dy(1) = x(2);
dy(2) = 3(1-x(1)  x(1))   x(2) - x(1);
```

The following statements integrate the van der Pol equation over the interval [0,10] with zero initial conditions.

```
>>  x0=zeros(2,1);
>>  [t,x] = ode45('vandP', 0, 10, x0);
```

1.4.b Nonlinear Algebraic Equations

MATLAB provides a several functions that perform optimization-related works. In this section, MATLAB functions solving the minimizing functions and finding zeros of functions are discussed.

Minimization of a function
The MATLAB command *fmin* minimizes a function by finding over a value which minimizes the given function. The syntax is shown below.

```
sol = fmin('funct', x1, x2)
```

where 'funct' is the name of a function to be minimized, x1 and x2 are lower limit and

upper limit of the interval of the function argument, respectively. For example, let us consider the following problem: Find the solution that minimizes the following performance index.

$$J = x_1^2 + \frac{(x_1 x_2 - 3)^2}{x_2}$$

For the given example, function *perf.m* should be first defined as follows.

```
function  J = perf(x)
J = x(1)  x(1) + ( x(1)  x(2))^2/x(2);
```

The minimum solution between –3 and 3 can be obtained by the following command.

```
>> sol = fmin('perf', -3, 3)
```

The *fmins* function is similar to *fmin* except that given function is a function of several variables, and an initial vector x0 should be specified rather than an interval. The syntax is shown below.

```
Sol = fmins('funct', x0)
```

Note that the *fmins* produces a local miminum solution.

Finding zeros of a function
The MATLAB command *fzero* tries to find a zero of one variable. The syntax is shown below.

```
Sol = fzero('funct', x0)
```

where 'funct' is the name of a function and x0 is an initial value of the scalar variable. The *fzero* function first searches for an interval around the given initial value where the function changes sign. The solution returned by *fzero* is near a point where 'funct' changes sign, or NaN if the search fails. Note that x0 can be a vector of length 2. In this case, x0 is an interval where the sign of funct(x0(1)) differs from the sign of funct(x0(2)). An error occurs if this is not true. Calling *fzero* with an interval guarantees *fzero* will return a value near a point where the function changes sign.

1.4.c Numerical Integration (Quadrature)

The integral of a function *f(x)* over the interval [a,b] is defined to be the area under the curve *f(x)* between a and b, which can be determined by numerically integrating *f(x)*, a process referred to as *quadrature*. MATLAB has two quadrature functions for performing numerical function integration as follows:

quad	Adaptive Simpson's rule
quad8	Adaptive Newton Cotes 8 panel rule

The syntax is shown below.

```
>> q = quad('funct', a, b)
```

where 'funct' is the name of a function, a and b are interval to be integrated.

Chapter 2
Linear Control Systems

Before designing a control system for a given system, the modeling procedure of the given system is required to analyze the dynamic characteristics of the system. Usually, a mathematical model of a dynamic system is defined as a set of nonlinear partial differential equations. The mathematical model includes unmodeled dynamics as well as uncertainties, i.e., the physical system cannot be completely modeled. By discretizing the spatial coordinates, the partial differential equations can be approximated by the ordinary differential equations. And by linearizing the system around the operating point or equilibrium state, linear modeling can be obtained. Even though the behavior physical systems are in general nonlinear with respect to inputs, most of physical systems can be approximated by linear systems as long as the system is operating around the equilibrium. A system is called linear if the principle of superposition applies. The linear characteristics make us analyze the system and design control systems very easily. MATLAB supports a lot of analysis tools as well as control system design tools for linear systems. In this chapter, the basic theories of linear systems will be briefly summarized, covering the stability analysis and control design methods.

2.1 Linear System Analysis

2.1.a Mathematical Representation

Consider a linear system represented by the state-space form:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1}$$

where $x \in R^n$ is state variables, $u \in R^m$ is control variables, and $y \in R^l$ is output variables. The state variable representation can be converted to the transfer function form:

$$\frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \tag{2}$$

where s denotes the Laplace variable.

MATLAB has useful commands to transform a mathematical model of linear system to another model. The command *tf2ss* converts the system in the transfer function form to the state-space form as follows:

```
>>  [A,B,C,D]=tf2ss(num,den)
```

where *num* and *den* are numerator and denominator of the transfer function, respectively. If the system has one input and one output, then the command *ss2tf* converts the system in state space to the transfer function as follows:

```
>>  [num, den]=ss2tf(A,B,C,D)
```

If the system has more than one input, the following command is used.

```
>>  [num, den]=ss2tf(A,B,C,D,iu)
```

where *iu* is an input index for transfer function.

2.1.b Stability of the system

Consider an autonomous system described by the linear vector differential equation

$$\dot{x} = Ax + Bu \tag{3}$$

The above system is said to be stable, if the solution of Eq. (3) tends toward zero (which is obviously the *only* equilibrium state if A is of full rank) as for an arbitrary initial condition. It is well known that the solution of Eq. (3) is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{4}$$

If all eigenvalues of A are distinct, the response of system ensuing from a given initial condition x_0 with zero control input can be written as

$$x(t) = e^{At}x(0) = \Phi e^{-\Delta t}\Phi^{-1}x(0) \tag{5}$$

where $\Delta = diag(\lambda_1, \cdots, \lambda_n)$, λ_i are the eigenvalues of A, futher, Φ denotes the modal matrices of A. For the repeated eigenvalue case, the situation is much more complicated (i.e., we should solve for the generalized eigenvectors of A). The generalization of Eq. (5) for the case of generalized eigenvectors has a similar form but will not be discussed here

From Eq. (5), we can see that the system will be asymptotically stable if and only if all the eigenvalues of A have negative real parts, i.e.,

$$\{\lambda_i(A)\} < 0 \tag{6}$$

where $\{\}$ denotes the real part of $\{\}$. Thus, the stability of a linear system can be completely characterized by the eigenvalues of the system. This eigenvalue approach to stability yields both necessary and sufficient conditions.

2.1.c Controllability and Observability

The purpose of this section is to introduce concepts of controllability and observability. The concepts of controllability and observability are very important in the study of dynamical system control and estimation problems. The concept of controllability is closely related to the existence of a feasible control law, since it is not always possible to find a control law of a given form that makes the closed-loop system stable with respect to the desired state or trajectory. Similarly, the concept of observability is related to the existence of a feasible algorithm for estimating the state variables from measurements. Controllability measures the particular actuator input configuration's ability to control all system states, whereas observability measures the particular sensor output configuration's ability to obtain the information needed to

estimate all system states. Following are the conventional definitions for linear dynamical systems.

Consider a linear system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{7}$$

Definition: Controllability
A linear system is said to be *controllable* at time t_0 if there exists a control $u(t)$ that will transfer the initial state $x(t_0)$ to the origin at some finite time $t_1 > t_0$. If this is true for *all initial times* t_0 and *all initial states* $x(t_0)$, the system is said to be *completely controllable*.

There are several necessary and sufficient conditions (or criteria) for Controllability.

Theorem: Criteria for Controllability
A system is completely controllable if and only if any one of the following equivalent conditions is satisfied:
(1) The $n \times nm$ controllability matrix C has rank n.

$$C \equiv \begin{bmatrix} B & AB & A^2B & \cdots & A^nB \end{bmatrix} \tag{8}$$

(2) The controllability grammian W_c is full-rank (in fact, positive definite).

$$W_c \equiv \int_0^\infty e^{A\tau} BB^T e^{A^T\tau} d\tau \tag{9}$$

There are other criteria for checking the controllability of the linear system, known as PBH(Popov, Belevitch, and Hautus) rank test and PBH eigenvector test that will not be discussed in this book.

Note that MATLAB operators for computing controllability matrix C and the controllability grammian W_c are as follows:

```
>> C = ctrb(A,B)
>> Wc = gram(A,B)
```

Definition: Observability

A linear system is said to be *observable* at t_0, if $x(t_0)$ can be determined from the knowledge of $u(\tau)$, $y(\tau)$ for $t_0 \le \tau \le t_1$. If this is true for all t_0 and all $x(t_0)$, the system is said to be *completely observable*.

Theorem: Criteria for Observability

A system is completely observable if and only if any one of the following equivalent conditions is satisfied:

(1) The $nl \times n$ observability matrix O has rank n.

$$O \equiv \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix} \tag{10}$$

(2) The observability grammian W_o is full-rank (in fact, positive definite).

$$W_o \equiv \int_0^\infty e^{A^T\tau} C^T C e^{A\tau} d\tau \tag{11}$$

There are also PBH tests for checking the observability of the linear system that will not be discussed in this book. The above criteria show the *duality* of the observability and controllability. It has been proven that the controllability and observability grammian matrices satisfy the following Lyapunov equations:

$$\begin{aligned} AW_c + W_c A^T + BB^T &= 0 \\ A^T W_o + W_o A + C^T C &= 0 \end{aligned} \tag{12}$$

Note that MATLAB operators for computing the observability matrix O and the observability grammian W_o are as follows:

```
>> O = obsv(A,B)
>> Wo = gram(A',C')
```

2.1.d Eigenvalue Problem

Let us introduce the generalized eigenvalue problem. Consider the system of differential equations

$$B\dot{x} = Ax \tag{13}$$

Note that overdots denote time differentiation throughout this text. The right and left eigenvalue problems associated with $x = \varphi e^{\lambda t}$ solutions of Eq. (13) are

$$\begin{aligned} &\textit{right}: && \lambda_i B\varphi_i = A\varphi_i, \\ &\textit{left}: && \lambda_i B^T\psi_i = A^T\psi_i, \end{aligned} \qquad i = 1,2,\cdots,n \tag{14}$$

with the usual normalization of the biorthogonality conditions

$$\begin{aligned} &\varphi_i^T B\varphi_i = 1, && i = 1,2,\cdots,n \\ &\psi_j^T B\varphi_i = \delta_{ij}, && i,j = 1,2,\cdots,n \end{aligned} \tag{15}$$

so that

$$\psi_j^T A\varphi_i = \lambda_i\delta_{ij}, \qquad i,j = 1,2,\cdots,n \tag{16}$$

The MATLAB statement to solve this eigenvalue problem is

```
>> [Phi, Lambda] = eig(A, B)
>> [Psi, Lambda] = eig(A',B')
```

where Phi and Psi are right and left *modal matrices*, and Lambda is a matrix with the eigenvalues on the diagonal.

Note that every eigenvalue solver (computer software package) uses its own special normalization equation for eigenvectors. In MATLAB, eigenvectors are normalized so that norm of each eigenvector becomes 1. In order to obtain eigenvectors that

satisfy some other normalization conditions [e.g., Eq. (15)], we need our own special normalization routine. The following code solves right and left eigenvalue problems and normalizes eigenvectors according to Eq. (15). The MATLAB operator to compute this normalized eigensolution is

```
>> [Lambda, Phi, Psi]=eign(A,B)
```

Observe that the above code cannot be used for the repeated eigenvalue case since eigenvectors satisfying Eqs. (15) and (16) do not generally exist. The source code for *eign* is shown in Table 2.1.

There are many eigenvalue problem-related applications, such as system analysis and synthesis techniques that use modal frequencies and mode shapes, as well as stability analysis and controller design methods such as eigenstructure assignment techniques. We will consider these in subsequent chapters of this text.

Now let us introduce *spectral decomposition*. Defining the modal matrices of right and left eigenvectors as

$$\begin{aligned} \Phi &= \begin{bmatrix} \varphi_1, & \cdots & \varphi_n \end{bmatrix} \\ \Psi &= \begin{bmatrix} \psi_1, & \cdots & \psi_n \end{bmatrix} \end{aligned} \tag{17}$$

For the case that B matrix is an identity matrix, $\Phi^{-1} = \Psi^T$ from Eq. (15). Therefore, we can obtain spectral decomposition of matrix A as follows:

$$A = \Phi \Delta \Psi^T = \Phi \Delta \Phi^{-1} \tag{18}$$

The diagonalization of A and existence of a nonsingular Φ depend upon matrix A having n linearly independent (but not necessarily unique) eigenvectors. If the eigenvalues of A are distinct, A can be diagonalized, otherwise we generally must solve a more complicated generalized eigenvalue problem for repeated eigenvalues for obtaining the modal matrix Φ, and Δ takes on a form that generally has entries only on the diagonal and the superdiagonal.

2.1.e Eigenvalue Sensitivities

The usefulness of modal sensitivities for analysis and design of engineering systems is well known. Some specific applications include identification of dynamical systems, optimum design of structural systems, and design of control systems by pole placement. In the above algorithms, eigenvalue and eigenvector derivatives with respect to design parameters are often useful. In this section, for the nondefective linear systems, analytic expressions for eigenvalue and derivatives are derived using a modal expansion approach.

Consider the nondefective system

$$B\dot{x} = Ax \tag{19}$$

The right and left eigenvalue problems of the above system are represented as follows.

$$\lambda_i B\varphi_i = A\varphi_i, \qquad i = 1,2,\cdots,n \tag{20}$$

$$\lambda_i B^T\psi_i = A^T\psi_i, \qquad i = 1,2,\cdots,n \tag{21}$$

where the conventional normalization of the biorthogonality conditions for the eigenvectors are adopted as in Eqs. (15) and (16).

Theorem: Eigenvalue Sensitivity

Partial derivative of eigenvalues of the system (19) is given by the following equation:

$$\frac{\partial\lambda_i}{\partial\rho} = \psi_i^T\left(\frac{\partial A}{\partial\rho} - \lambda_i\frac{\partial B}{\partial\rho}\right)\varphi_i \tag{22}$$

where ρ is some arbitrary parameter upon which A and B depend.

Proof:

Let us derive the eigenvalue derivatives with respect to a scalar parameter ρ. First, we take the partial derivative of Eq. (20) to get

$$\frac{\partial \lambda_i}{\partial \rho} B\varphi_i + \lambda_i \frac{\partial B}{\partial \rho}\varphi_i + \lambda_i B\frac{\partial \varphi_i}{\partial \rho} = \frac{\partial A}{\partial \rho}\varphi_i + A\frac{\partial \varphi_i}{\partial \rho} \tag{23}$$

We then premultiply the above equation by ψ_j^T to obtain

$$\frac{\partial \lambda_i}{\partial \rho}\psi_j^T B\varphi_i + \left(\lambda_i \psi_j^T B - \psi_j^T A\right)\frac{\partial \varphi_i}{\partial \rho} = \psi_j^T\left(\frac{\partial A}{\partial \rho} - \lambda_i \frac{\partial B}{\partial \rho}\right)\varphi_i \tag{24}$$

Using Eqs. (15) and (21) in the above equation gives the analytical expressions of eigenvalue sensitivities as

$$\frac{\partial \lambda_i}{\partial \rho} = \psi_i^T\left(\frac{\partial A}{\partial \rho} - \lambda_i \frac{\partial B}{\partial \rho}\right)\varphi_i \tag{25}$$

2.1.f State Covariance Matrix

Consider a linear system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{26}$$

In general, the state response may describe the detailed information about system performance. And the state covariance matrix gives useful information to study the properties of linear system, because certain information about the linear system is embodied in the state covariance matrix. The state covariance matrix due to impulsive inputs in u is defined as follows:

$$X \equiv \int_0^\infty x(t)x^T(t)dt \tag{27}$$

Using Eq. (4), the response of system, i.e., the solution of Eq. (26), ensuing from impulsive inputs with zero initial condition can be written as

$$x(t) = e^{At}B \tag{28}$$

Substitution of Eq. (28) into Eq. (27) yields the state covariance matrix as follows:

$$X \equiv \int_0^\infty e^{At} BB^T e^{A^T t} dt \tag{29}$$

The state covariance matrix X can be computed by solving Lyapunov equation introduced in the following theorem.

Theorem: State Covariance Matrix for Linear Systems
The state covariance matrix satisfies the following Lyapunov equation if and only if a linear system is asymptotically stable.

$$AX + XA^T + BB^T = 0 \tag{30}$$

Proof:
(Necessity)
The integrand is an infinite sum of terms of the form $t^k e^{\lambda t}$, where λ is an eigenvalue of A and, since $\{\lambda_i(A)\} < 0$, the integrand will exist for all values of t. Next, by direct substitution of Eq. (29) into the left hand side of Eq. (30), we find that the combined integrand is a perfect differential, permitting us to directly establish

$$\begin{aligned} AX + XA^T &= \int_0^\infty Ae^{At} BB^T e^{A^T t} dt + \int_0^\infty e^{At} BB^T e^{A^T t} A^T dt \\ &= \int_0^\infty \frac{d}{dt}\left(e^{At} BB^T e^{A^T t}\right) dt = e^{At} BB^T e^{A^T t}\Big|_0^\infty \\ &= -BB^T \end{aligned} \tag{31}$$

Note that e^{At} (and $e^{A^T t}$) $\to 0$ as $t \to \infty$, since all eigenvalues of A have negative real parts, and since e^{At} (and $e^{A^T t}$) equal 1 at $t = 0$.

(Sufficiency)
The sufficiency part can be proved by Lyapnov (direct) stability theorem that will not be covered in this book.

Note that the Lyapunov equation is equivalent to a set of $n(n+1)/2$ linear equations in $n(n+1)/2$ unknowns for an nth order system. The Lyapunov equation can be solved by using numerical algorithms utilizing QR factorization; Schur decomposition, or spectral decomposition, however, we believe that the most efficient and robust algorithm utilizes the QR factorization. Note that the MATLAB operators for computing the solution of Lyapunov equations are

```
>> X = lyap(A, B  B')
```

2.2 Linear Optimal Control

Among the most fundamental results in modern control theory are the two sets of analytical developments that underlie the linear-quadratic regulator (LQR) and eigenstructure assignment regulator (EAR). Design of control systems for aeroservoelasticity problems has been accomplished using both design techniques. The central aspects of these two sets of approaches are presented in this section.

In the LQR approach, the central feature is the minimization of a quadratic performance index, subject to satisfaction of a linearized system differential equation model. The integral quadratic performance index appeals to the intuitive energy-based notion that we should seek a weighted compromise between the closed-loop output error energy and a measure of the control effort required to regulate the system. However, the fundamental practical difficulties in applying the LQR approach arise because it is difficult to capture many practical performance, robustness, and admissible design inequality constraint issues in a single scalar performance measure, simply through a judicious a priori choice of weight matrices.

Since the complete response of a given linear system can be fully determined by the closed-loop eigensolution, another attractive approach is to find a control law that satisfies prescribed constraints on the closed-loop eigenvalues and eigenvectors. It is also possible to determine many of the candidate measures of performance and stability robustness (insensitivity with respect to system model errors and external disturbances) as simple functions of the closed-loop eigenvalues and eigenvectors. The class of methods that directly impose prescribed constraints on the eigenvalues

and eigenvectors are known as *eigenstructure assignment regulator* design methods. While the EAR approaches allow some immediate insight that narrows the search for practical designs, some additional difficulties arise. For controllable systems, we will see that the closed-loop eigenvalues can be placed exactly at an infinity of locations while, generally, the closed-loop eigenvectors can be placed only in certain feasible subspaces.

The quadratic regulator and eigenstructure assignment approaches are among the most important representatives of modern control theory, which is essentially the set of approaches based on the time-domain state space description of linear dynamical systems. As will be evident in the discussion below, the EAR formulations are actually a diverse family of recently developed concepts and the associated gain design algorithms that subsume and greatly expand upon the methodology historically known as *pole placement*.

2.2.a Linear Quadratic Regulator

If a dynamical system is *completely controllable*, these exists at least one control that will transfer the system from any/all given initial states to any/all desired states. Consider a completely controllable and observable dynamical system whose behavior is modeled by a nonlinear system of differential equations in the first-order state variable form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{32}$$

where $x(t) \quad R^n$ is the state vector, and $u(t) \quad R^m$ is the control vector.

The objective of optimal control theory is to determine the control law that satisfies the differential equation constraints, i.e., all trajectories are consistent with our model of the system dynamics [Eq. (32)], and at the same time minimizes (or maximizes) some performance criterion. The most general performance index functional to be minimized (in the current discusion) is defined as

$$J = h\left(x\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} g\left(x(t), u(t), t\right)\, dt \tag{33}$$

where h and g are given functions, t_0 is the initial time, and t_f is the final time. The objective is to find the optimal control $u(t)$ that drive the solution of the differential equations of Eq. (32) from the given initial conditions to the final time such that the performance index of Eq. (33) is minimized.

The derivation of the necessary conditions for optimal control is relatively lengthy and straightforward and is not presented here. The linear regulator problem is the most obvious special case of a linear system with the feedback control law being parameterized as a linear function of the instantaneous state. The object is to determine an optimal feedback control gain K for a linear law of the form

$$u(t) = -Kx(t) \tag{34}$$

which minimizes the quadratic performance index

$$J = \frac{1}{2} x^T(t_f) S x(t_f) + \frac{1}{2} \int_{t_0}^{t_f} \left\{ x^T(t) Q x(t) + u^T(t) R u(t) \right\} dt \tag{35}$$

where S and Q are real symmetric positive semidefinite matrices, and R is a real symmetric positive definite matrix. It is assumed that the states and controls are not bounded. Moreover, we assume that all elements of the final state vector $x(t_f)$ are free and the final time t_f is fixed.

Note that the physical and geometric motivation for this performance measure is as follows: It is desired to maintain the time-varying state close to the origin without incurring an excessive expenditure of control effort. The balance between small errors and small control effort is achieved through judicious assignment of the weight matrices.

By applying the principles of the calculus of variations to minimization of the performance index, the necessary conditions for the linear quadratic regulator problem are obtained: the matrix differential Riccati equation.

$$\frac{dP(t)}{dt} = -A^T P(t) - P(t) A + P(t) B R^{-1} B^T P(t) - Q \tag{36}$$

The solution $P(t)$ (an $n \times n$ positive semi-definite symmetric matrix), called the Riccati matrix, is a symmetric positive definite matrix. The optimal gain matrix is related to the Riccati gains in the following manner:

$$K(t) = R^{-1}B^{T}P(t) \tag{37}$$

Note that, even though the plant matrices are constant, the optimal feedback gain matrix K is time-varying for the finite time control case. Also, note that measurements (or estimates) of all of the state variables must be available to implement the optimal control law. For obvious reasons, such a law is known as a full state feedback control law.

The most important special case of the linear regulator problem is the one for which: (i) The plant is completely controllable, (ii) The plant dynamics are time invariant, i.e., A, B, Q, and R are constant matrices, (iii) The final state error weight matrix is zero, S=0 in Eq. (2.35), and (iv) The final time t_f is infinite. In other words, the problem is to find the optimal control minimizing the performance index

$$J = \frac{1}{2}\int_{t_0}^{\infty}\left\{x^{T}(t)Qx(t) + u^{T}(t)Ru(t)\right\}\,dt \tag{38}$$

In this case, it can be established that the optimal control law for minimizing the quadratic index over an infinite time duration is stationary, and the feedback gain matrix is a constant,,

$$\begin{aligned} u(t) &= -Kx(t) \\ K &= R^{-1}B^{T}P \end{aligned} \tag{39}$$

determined by the constant solution P of the matrix Riccati equation, Eq. (36), reduced to a set of quadratically nonlinear algebraic equations,

$$A^{T}P + PA - PBR^{-1}B^{T}P + Q = 0 \tag{40}$$

which is called the *algebraic Riccati equation* (ARE). The fact that P approaches a constant can be established from the general time-varying $P(t)$ solution using numerical method and letting $t_f \to \infty$.

Note that P in Eq. (40) is an $n \times n$ matrix, carrying out the algebra to obtain the scalar elements of Eq. (40) generates a coupled system of n^2 first-order differential equations. Notice that if P satisfies Eq. (40), P^T also satisfies it, because the transpose of Eq. (40) yields exactly Eq. (40). Since P is symmetric, there are not n^2 but rather $n(n+2)/$ equations that must be solved.

Note that the MATLAB (Control System Toolbox) operator for the steady-state LQR problem has the following syntax:

```
>> [K,P] = lqr(A, B, Q, R)
```

which returns the constant optimal gain K and the ARE solution P.

2.2.b Linear Quadratic Regulator with Output Feedback

In the previous section, full state optimal control law was discussed. In real situation, only some of the states are available as measured outputs. Therefore, in order to utilize the full state feedback, observer or estimator should be implemented with controller to estimate the state variables using the mathematical model and output measurements. This makes the control system very complicated. In this section, the LQR with output feedback is briefly introduced. Because the estimator is not involved, the whole control system using output feedback control law will become very simple. However, the control system performance will be degraded because only limited information is used for output feedback purpose, and the design procedure will be more complicated than the full state LQR.

Consider a linear time-invariant system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{41}$$

The control law is restricted to a linear time-invariant output feedback of the form

$$u = -Ky \tag{42}$$

where K is an output feedback gain matrix to be determined by the design procedure. The design objective for flutter suppression may be obtained by selecting the control input to minimize a quadratic performance index of the type

$$J = \frac{1}{2}\int_{t_0}^{\infty} \left\{ y^T(t)Qy(t) + u^T(t)Ru(t) \right\} dt \tag{43}$$

where $Q \geq 0$ and $R > 0$ are symmetric weighting matrices.

The performance index may be expressed in terms of K as

$$J = \frac{1}{2}\int_{0}^{\infty} x^T \left(C^T Q C + C^T K^T R K C \right) x \, dt \tag{44}$$

The design problem is to select the gain K so that J is minimized subject to following the system dynamic constraints

$$\dot{x} = (A - BKC)x \equiv A_c x \tag{45}$$

The dynamical optimization problem may be converted into an equivalent static optimization problem that is easier to solve. Assume that there exists a constant, symmetric, positive semidefinite matrix P that satisfies the following equation.

$$\frac{d}{dt}\left(x^T P x \right) = -x^T \left(C^T Q C + C^T K^T R K C \right) x \tag{46}$$

By assuming that the closed-loop system is asymptotically stable so that $x(t)$ vanishes with time, the performance index can be represented as

$$J = \frac{1}{2} x^T(0) P x(0) = \frac{1}{2} trace\{ P x(0) x^T(0) \} \tag{47}$$

where the operator $trace(\ )$ denotes the trace of a square matrix. If P satisfies Eq. (46), then we may use Eq. (45) to see that

$$-x^T\left(C^TQC + C^TK^TRKC\right)x = \frac{d}{dt}\left(x^TPx\right) = \dot{x}^TPx + x^TP\dot{x} \\ = x^T\left(A_c^TP + PA_c\right)x \tag{48}$$

The above equation should satisfy for all initial conditions, and therefore the following condition should be satisfied for all state trajectories.

$$A_c^TP + PA_C + C^TK^TRKC + C^TQC = 0 \tag{49}$$

Now we can state that, if there exists a constant symmetric positive semidefinite matrix P that satisfies Eq. (49), then the performance index J for the closed-loop system can be represented in terms of P by Eq. (47). Since the initial state $x(0)$ may not be known, we adopt Athans and Levine's procedure that minimizes the expected value of performance index

$$\bar{J} = E\{J\} = \frac{1}{2}\,trace(PX) \tag{50}$$

where the operator $E(\ )$ denotes the expected value over all possible initial conditions, and the $n \times n$ symmetric matrix X is defined by

$$X \equiv x(0)x^T(0) \tag{51}$$

The usual assumption is that the initial states are uniformly distributed on the unit sphere; then X becomes identity matrix. By applying optimality conditions for the above problem, the optimal control gain matrix K can be obtained by solving the following three coupled nonlinear algebraic matrix equations simultaneously.

$$A_c^TP + PA_C + C^TK^TRKC + C^TQC = 0 \tag{52}$$

$$A_CS + SA_c^T + X = 0 \tag{53}$$

$$RKCSC^T - B^TPSC^T = 0 \tag{54}$$

Note that P, S, and K are matrices to be determined from the above three equations. There exists several numerical approaches to solve Eqs. (52)-(54). Gradient-based routines can be used with various optimization algorithms. Moerder and Calise proposed an efficient iterative algorithm, which converges to a local minimum if the following conditions hold: (i) there exists an output gain matrix K that A_c is stable, (ii) the output matrix C has full row rank, (iii) control weighting matrix R is positive definite, and (iv) output state weighting matrix Q is positive semidefinite and should be chosen so that all unstable states are weighted in the performance index. The iterative algorithm is outlined as follows:

Step 1: Choose K such that the closed-loop system matrix A_c has all eigenvalues with negative real parts.

Step 2: With the given K and the according A_c, solve the Lyapunov Eqs. (52) and (53) for P and S, and compute the performance index value $\bar{J}$.

Step 3: Evaluate the gain update direction

$$\Delta K = R^{-1}B^T PSC^T\left(CSC^T\right)^{-1} - K \tag{55}$$

Note that for the case that the matrix CSC^T in Eq. (55) is singular, Moore-Penrose generalized inverse utilizing singular value decomposition may be useful to inverse the matrix CSC^T to deal with the rank deficiency problem

Step 4: Update the gain by

$$K_{new} = K + \alpha\Delta K \tag{56}$$

where α is chosen so that the closed-loop system A_c is asymptotically stable, and the performance index of the updated system is less than that of the current system. To obtain the appropriate α, set the initial step size α_0 (for example, 0.3), and proceed if the above two conditions are satisfied. If not, decrease the step size using the following simple rule till the above conditions are satisfied.

$$\alpha_{new} = \kappa\alpha_{old}, \qquad 0 < \kappa < 1 \tag{57}$$

Step 5: Iterate Steps 2 through 4 until converged solution is obtained.

Any minimization algorithm requires the selection of an initial stabilizing gain, however such a gain may or may not be easy to find. One possible way to find the initial gain matrix is to use root locus techniques by closing one loop at a time in the control system. For the initial gain matrix for the iterative procedure, we first solve the full state LQR problem with performance index Eq. (2.42), and construct an initial stabilizing output gain matrix by choosing the corresponding gain parameters to be used in the output feedback out of the resulting LQR gain matrix elements. Let us explain this procedure in detail. The following relation can be obtained by considering the full state feedback and output feedback control scheme

$$u = -K_F x = -KCx \tag{58}$$

where K_F is the full state gain matrix resulting from LQR solver. The initial output feedback gain matrix can be computed using the following equation.

$$K_F = KC \tag{59}$$

In general, the matrix C is not square, and therefore the Moore-Penrose pseudo-inverse should be used as follows:

$$K = K_F C^+ \tag{60}$$

where the pseudo-inverse of the matrix C can be expressed as

$$C^+ = V_1 \Sigma^{-1} U_1^T \tag{61}$$

where U_1, V_1 are unitary matrices obtained by singular values decomposition of the matrix C, $\Sigma = diag\{\sigma_1, \cdots, \sigma_r\}$ contains as its elements the real singular values of C, and r denotes the rank of C.

2.3 Pole Placement Technique

2.3.a Sylvester Approach

The eigenstructure assignment regulator theory is employed to regulate the system with the smallest control effort. It is assumed that all states are available for feedback. The gain matrix of the completely controllable and observable linear dynamic is expressed as follows

$$u = -Kx \tag{62}$$

The right and left eigenvalue problems of the closed-loop system can be written, respectively, as

$$(A - BK)\varphi_i^c = \lambda_i^c \varphi_i^c \tag{63}$$

where φ_i^c is the right eigenvectors of the closed-loop system corresponding to the closed-loop eigenvalue λ_i^c. The pole placement algorithm based on Sylvester equation utilizes the parameter vector defined by

$$h_i = K\varphi_i^c \tag{64}$$

Substituting Eq. (64) into Eq. (63) yields the following Sylvester equation

$$(A - \lambda_i^c I)\varphi_i^c = Bh_i \tag{65}$$

or in matrix form

$$A\Phi_c - \Phi_c \Lambda_D = BH \tag{66}$$

where

$$\begin{aligned} \Phi_c &= [\varphi_1^c, \varphi_2^c, ..., \varphi_n^c] \\ \Lambda_D &= diag(\lambda_1^c, \lambda_2^c, ..., \lambda_n^c) \\ H &= [h_1, h_2, \ldots h_n] \end{aligned} \tag{67}$$

with Λ_D containing the desired closed-loop eigenvalues. From Eq. (65), it can be seen that

$$\varphi_i^c = (A - \lambda_i^c I)^{-1} B h_i \tag{67}$$

If λ_i^c are distinct from their open-loop positions, the columns of H generate the corresponding closed-loop eigenvectors. For the given parameter matrix H, the closed-loop modal matrix Φ_c can be determined by solving Sylvester equation. The gain matrix is determined using the relation Eq. (64)

$$K = H\Phi_c^{-1} \tag{68}$$

2.3.b Nonlinear Programming (NLP) Approach

The eigenvalues of the system can be assigned to the desired values by using a pole placement technique. Our problem can be stated as the nonlinear parameter optimization problem in which we seek to impose specified eigenspace equality constraints, and the elements of output feedback gain matrix are parameters to be determined. We introduce the output feedback control law

$$u = -Ky \tag{69}$$

where K denotes the output feedback gain and y includes only the measurement information. The eigenvalue problem of the closed-loop system can be written as

$$(A - BKC)\varphi_i^c = \lambda_i^c \varphi_i^c \tag{70}$$

where φ_i^c is the eigenvectors of the closed-loop system corresponding to the closed-loop eigenvalue λ_i^c. Let us define p as our parameter vector that consists of the elements of output feedback gain K. Now, we can formulate our nonlinear programming problem as follows:

Find output feedback gain parameter p satisfying the following equality constraint equation that is related to the closed-loop eigenvalue assignment

$$f_i(p) = \lambda_i^d - \lambda_i(p) = 0, \qquad i = 1, 2, .., r \tag{71}$$

where r is the number of the structural modes, and λ_i^d denotes the desired eigenvalues of the structural modes. The above problem can be solved by using any gradient-based nonlinear programming algorithm. In this section, homotopic nonlinear programming with minimum norm correction algorithm is used. In order to enhance the convergence, the linear homotopy map is generated: original eigenvalue assignment problem is replaced by the one-parameter α family as follows:

$$f_i(p) = \left\{\alpha\lambda_i^d + (1-\alpha)\lambda_i(p_{start})\right\} - \lambda_i(p) = 0, \qquad 0 \le \alpha \le 1 \tag{72}$$

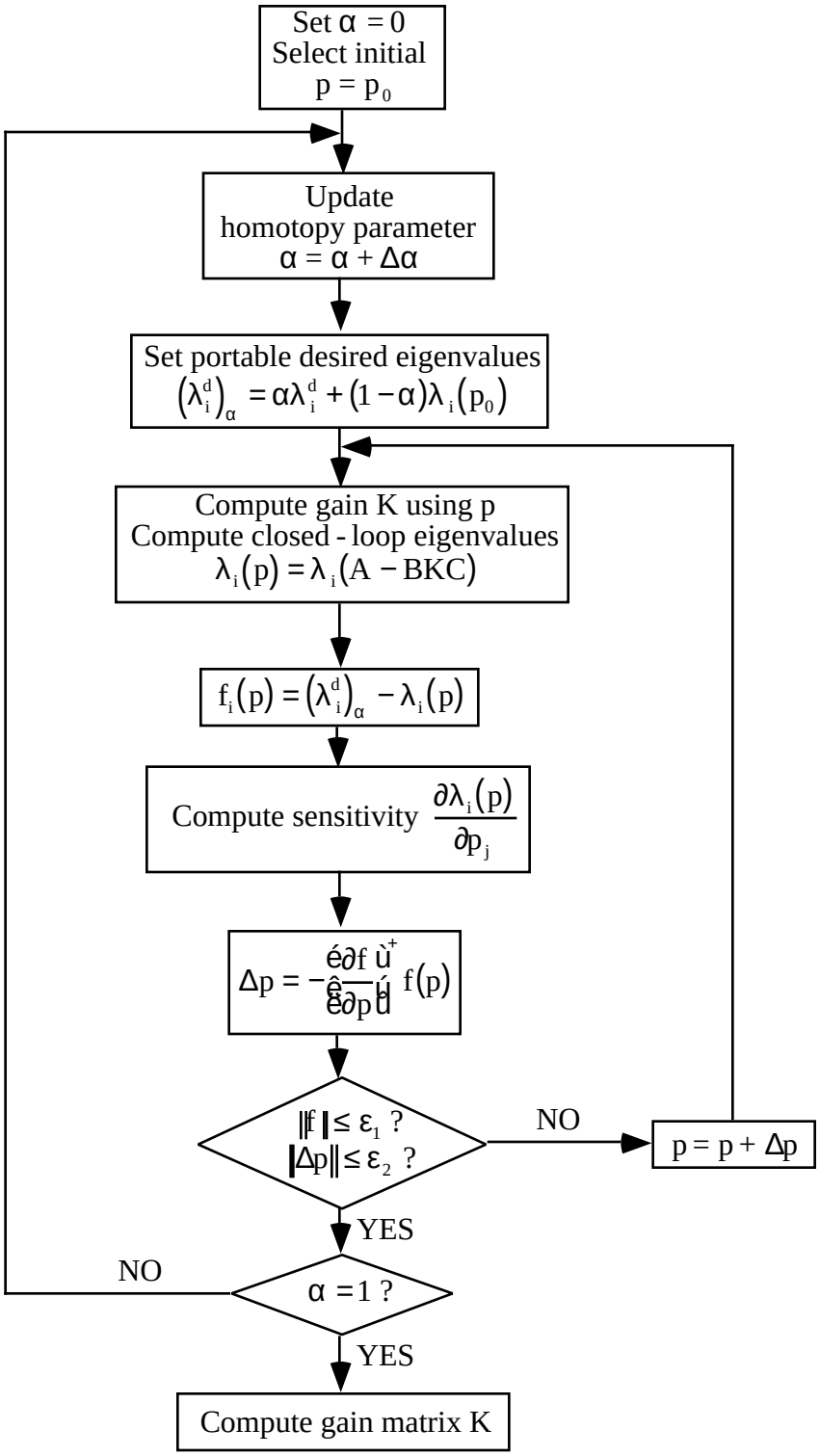
where p_{start} is the initial guess of parameter vector. Sweeping α at a suitably small increment generates a sequence of neighboring problems. We solve this sequence of problems using the neighboring converged solutions to generate starting iteratives for each subsequent problem. If we obtain a solution for $\alpha = 1$, then we have the desired solution. The local design corrections for each α are performed by linearizing the neighboring problem (nonlinear constraint equations) about the local solution and computing the minimum norm differential correction that satisfies the linearized constraint equation. This algorithm is a generalized Newton process for solving a system of underdetermined nonlinear equations. On each iteration, the norm of the correction vector is minimized while satisfying the following linearized constraint equation.

$$\left[\frac{\partial f}{\partial p}\right]_p \Delta p = -f(p) \tag{73}$$

where $f(p) = \left[f_1(p), f_2(p), ..., f_r(p)\right]^T$ is a constraint vector. The solution of the above equation can be obtained by pseudo-inverse of matrix $\left\{\partial f / \partial p\right\}\big|_p$ utilizing singular value decomposition. The eigenvalue sensitivities required in the above equation for iterative direction search can be evaluated by using the following analytic formulation.

$$\frac{\partial \lambda_i^c}{\partial p_k} = \psi_i^T \left(\frac{\partial (A - BKC)}{\partial p_k} \right) \varphi_i^c \tag{74}$$

where φ_i^c and ψ_i^c are normalized right and left eigenvectors of the closed-loop system corresponding to the eigenvalue λ_i^c, respectively. Figure shows the flow chart of this algorithm.


Set $\alpha = 0$
Select initial
$p = p_0$
Update
homotopy parameter
$\alpha = \alpha + \Delta\alpha$
Set portable desired eigenvalues
$(\lambda_i^d)_\alpha = \alpha\lambda_i^d + (1-\alpha)\lambda_i(p_0)$
Compute gain K using p
Compute closed - loop eigenvalues
$\lambda_i(p) = \lambda_i(A - BKC)$
$f_i(p) = (\lambda_i^d)_\alpha - \lambda_i(p)$
Compute sensitivity $\frac{\partial\lambda_i(p)}{\partial p_j}$
$\Delta p = -\left[\frac{\partial f}{\partial p}\right]^+ f(p)$
$\|f\| \le \varepsilon_1$?
$\|\Delta p\| \le \varepsilon_2$?
NO
$p = p + \Delta p$
YES
$\alpha = 1$?
NO
YES
Compute gain matrix K

Chapter 3

Aeroservoelastic Analysis of the Airfoil

3.1. Unsteady Aerodynamic Forces of the Typical Section Model

The unsteady aerodynamic forces are calculated based on the linearized thin-airfoil theory. In this section, Theodorsen's approach (*Ref: T. Theodorsen, "General Theory of Aerodynamic Instability and the Mechanism of Flutter", NACA TR 496, 1935*.) will be summarized and the flutter analysis will be conducted based on his approach.

In his approach, aerodynamic surfaces were modeled by flat plates. Theodorsen assumed that the flat airfoil is oscillating about the shear center (elastic axis) and unsteady flow is composed of two flow components, (a) non-circulatory flow which can be expressed through the sources and sinks and (b) circulatory flow related to the flat vorticity surface extending from trailing edge to the infinity. For each flow component, he obtained the velocity potential, then calculate the pressure using Bernoulli theory.

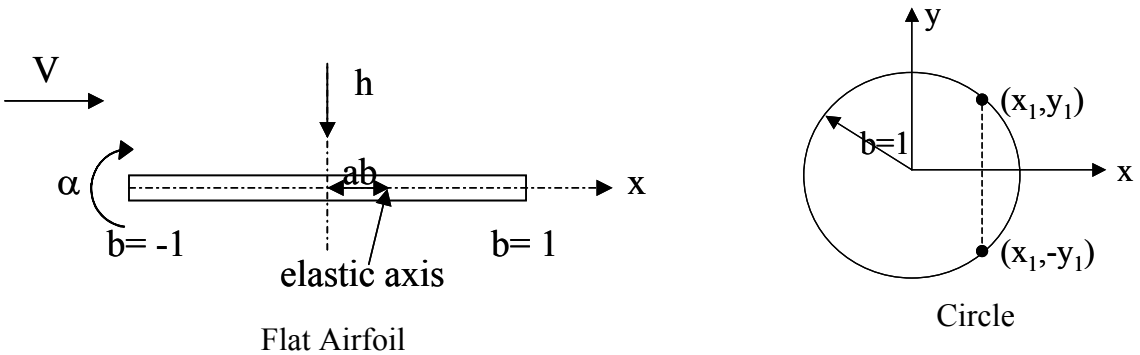


Figure . Flat airfoil and the transformed circle.

3.1.a The Non-circulatory Flow

By Joukowski's conformal transformation, the airfoil can be mapped onto a circle. The velocity potential of a source ε on circle (x_1, y_1) can be expressed as the follow.

$$\varphi = \frac{\varepsilon}{4\pi} Ln\left[(x - x_1)^2 + (y - y_1)^2\right] \tag{1}$$

Similarly, the velocity potential due to a source 2ε at on circle (x_1, y_1) and a sink -2ε at on circle $(x_1, -y_1)$

$$\varphi = \frac{\varepsilon}{2\pi} Ln\left[\frac{(x-x_1)^2 + (y-y_1)^2}{(x-x_1)^2 + (y+y_1)^2}\right] \tag{2}$$

Since $y = \sqrt{1-x^2}$, the velocity potential is a function of x only.

The downward displacement of the airfoil can be written as

$$z = h + \alpha(x - ab) \tag{3}$$

Then, the upwash will be

$$w_a(x,t) = -\left(\frac{\partial z}{\partial t} + V\frac{\partial z}{\partial x}\right) = -[\dot{h} + \dot{\alpha}(x-ab)] - V\alpha \tag{4}$$

Therefore, the velocity potential due to pitch angle, α will be

$$\varphi_\alpha = b\int_{-1}^{1}\varphi dx = b\int_{-1}^{1}\frac{-V\alpha}{2\pi} Ln\left[\frac{(x-x_1)^2 + (y-y_1)^2}{(x-x_1)^2 + (y+y_1)^2}\right]dx_1 = V\alpha b\sqrt{1-x^2} \tag{5}$$

Similarly, velocity potentials due to plunge motion, $\dot{h}$ and angular velocity, $\dot{\alpha}$ are respectively expressed as

$$\varphi_{\dot{h}} = \dot{h}b\sqrt{1-x^2} \tag{6}$$

$$\varphi_{\dot{\alpha}} = \frac{\dot{\alpha}b^2}{2\pi}\pi(x+2)\sqrt{1-x^2} - \dot{\alpha}b^2(1+a)\sqrt{1-x^2} = \dot{\alpha}b^2(\frac{x}{2}-a)\sqrt{1-x^2} \tag{7}$$

The total velocity potential due to noncirculatory flow becomes

$$\begin{aligned}\varphi_{NC} &= \varphi_\alpha + \varphi_{\dot{h}} + \varphi_{\dot{\alpha}} \\ &= V\alpha b\sqrt{1-x^2} + \dot{h}b\sqrt{1-x^2} + \dot{\alpha}b^2(\frac{x}{2}-a)\sqrt{1-x^2}\end{aligned} \tag{8}$$

By Bernoulli theorem, the pressure is obtained

$$\Delta p = -2\rho\left(\frac{\partial\varphi}{\partial t} + V\frac{\partial\varphi}{\partial x}\right) = -2\rho\frac{d\varphi}{dt} = -2\rho\dot{\varphi} \tag{9}$$

And the force (positive downward) and the pitching moment (positive nose-up) about the elastic axis will be expressed as

$$\begin{aligned} F_{NC} &= b\int_{-1}^{1}\Delta p\,dx = -2\rho b\int_{-1}^{1}\dot{\varphi}\,dx \\ &\equiv -\pi\rho b^2\left(\ddot{h} + V\dot{\alpha} - ba\ddot{\alpha}\right) \end{aligned} \tag{10}$$

$$\begin{aligned} M_{NC} &= b\int_{-1}^{1}\Delta p(x-a)b\,dx = -2\rho b^2\int_{-1}^{1}\frac{d\varphi}{dt}(x-a)dx \\ &\equiv \pi\rho b^2\left(V\dot{h} + ba\ddot{h} + V\dot{\alpha}^2 - b^2\left(\frac{1}{8}+a^2\right)\ddot{\alpha}\right) \end{aligned} \tag{11}$$

3.1.b The Circulatory Flow

To satisfy the Kutta condition, Theodorsen employed a bound vortex distribution over the airfoil, and a vortex distribution over the airfoil wake.

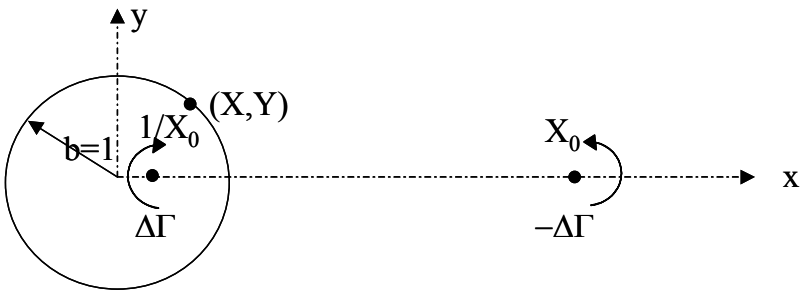


Figure Circle in circulatory flow

In order to consider wake, let's assume a bound vortex $\Delta\Gamma = \gamma dx$ at $1/X_0$, and a shed vortex $-\Delta\Gamma$ at X_0.

Then, the velocity potential due to a vortex

$$\begin{aligned} \Delta\varphi_\Gamma &= -\frac{\Delta\Gamma}{2\pi}\left[\tan^{-1}\frac{Y}{X-X_0} - \tan^{-1}\frac{Y}{X-1/X_0}\right] \\ &= -\frac{\Delta\Gamma}{2\pi}\tan^{-1}\left[\frac{\left(X_0 - 1/X_0\right)Y}{X^2 - X\left(X_0 + 1/X_0\right) + Y^2 + 1}\right] \end{aligned} \tag{12}$$

Let's define $X_0 + 1/X_0 = 2x_0$, and $X = x, Y = \sqrt{1-x^2}$

Then,

$$\begin{aligned} X_0 &= x_0 + \sqrt{x_0^{\,2} - 1} \\ 1/X_0 &= \frac{1}{x_0 + \sqrt{x_0^{\,2} - 1}} = x_0 - \sqrt{x_0^{\,2} - 1} \end{aligned} \tag{13}$$

The velocity potential can be expressed as

$$\begin{aligned} \Delta\varphi_\Gamma &= -\frac{\Delta\Gamma}{2\pi}\tan^{-1}\left[\frac{\sqrt{1-x^2}\left(2\sqrt{x_0^{\,2}-1}\right)}{x^2 - x(2x_0) + (1-x^2) + 1}\right] \\ &= -\frac{\Delta\Gamma}{2\pi}\tan^{-1}\left[\frac{\sqrt{1-x^2}\sqrt{x_0^{\,2}-1}}{x^2 - xx_0}\right] \end{aligned} \tag{14}$$

where

$$-1 \le x \le 1, \quad 1 \le x_0 \le \infty$$

It has to noted that the vortex is moving away from the airfoil with velocity of V. Therefore, by Bernoulli theorem, the pressure due to the vortex is

$$\Delta p = -2\rho\left(\frac{\partial\Delta\varphi_\Gamma}{\partial t} + V\frac{\partial\Delta\varphi_\Gamma}{\partial x}\right) \tag{15}$$

where

$$\begin{aligned} \frac{2\pi}{\Delta\Gamma}\frac{\partial\Delta\varphi_\Gamma}{\partial x} &= -\frac{\partial}{\partial x}\left[\tan^{-1}\left[\frac{\sqrt{1-x^2}\sqrt{x_0^{\,2}-1}}{1-xx_0}\right]\right] \\ &= \frac{\sqrt{x_0^{\,2}-1}}{\sqrt{1-x^2}}\frac{1}{x_0 - x} \end{aligned} \tag{16}$$

$$\frac{2\pi}{\Delta\Gamma}\frac{\partial\Delta\varphi_\Gamma}{\partial x_0}=\frac{\sqrt{1-x^2}}{\sqrt{x_0{}^2-1}}\frac{1}{x_0-x} \tag{17}$$

The pressure at x due to the vortex at x_0

$$\begin{aligned}\Delta p_\Gamma &= -2\rho\frac{\Delta\Gamma}{2\pi}V\left[\frac{\sqrt{x_0{}^2-1}}{\sqrt{1-x^2}}+\frac{\sqrt{1-x^2}}{\sqrt{x_0{}^2-1}}\right]\frac{1}{x_0-x}\\ &= -\rho V\frac{\Delta\Gamma}{2\pi}\left[\frac{x_0{}^2-x^2}{\sqrt{1-x^2}\sqrt{x_0{}^2-1}}\right]\frac{1}{x_0-x}\\ &= -\rho V\frac{\Delta\Gamma}{2\pi}\frac{x_0+x}{\sqrt{1-x^2}\sqrt{x_0{}^2-1}}\end{aligned} \tag{18}$$

The force on the whole airfoil due to a vortex at x_0 will be

$$\begin{aligned}\Delta F_\Gamma &= b\int_{-1}^{1}\Delta p_\Gamma dx\\ &= -\rho Vb\frac{x_0}{\sqrt{x_0{}^2-1}}\Delta\Gamma, 1\le x_0\le\infty\end{aligned} \tag{19}$$

The total force can be calculated by integrating with respect to x_0

$$\begin{aligned}F_\Gamma &= \int\Delta F_\Gamma\\ &= -\rho Vb\int_1^\infty\frac{x_0}{\sqrt{x_0{}^2-1}}\gamma dx_0\end{aligned} \tag{20}$$

Similarly,

$$\Delta M_\Gamma = b^2\int_{-1}^{1}\Delta p_\Gamma(x-a)dx, 1\le x_0\le\infty \tag{21}$$

$$\begin{aligned}M_\Gamma &= \int\Delta M_\Gamma\\ &= -\rho Vb^2\int_1^\infty\left[\frac{1}{2}\sqrt{\frac{x_0+1}{x_0-1}}-\left(a+\frac{1}{2}\right)\frac{x_0}{\sqrt{x_0{}^2-1}}\right]\gamma\, dx\end{aligned} \tag{22}$$

It has to be noted that the force and moment are function of vortex strength γ. By applying Kutta condition at trailing edge we can determine the vortex strength. First, let's get the total velocity potential. The total velocity potential is

$$\begin{aligned}\varphi_{\text{total}} &= \varphi_\Gamma + \varphi_\alpha + \varphi_{\dot{h}} + \varphi_{\dot{\alpha}} \\ &= \varphi_\Gamma + V\alpha b\sqrt{1-x^2} + \dot{h}b\sqrt{1-x^2} + \dot{\alpha}b^2(\frac{x}{2}-a)\sqrt{1-x^2}\end{aligned} \tag{23}$$

By applying the Kutta condition, we can get the following equation

$$\begin{aligned}&\frac{\partial\varphi_\Gamma}{\partial x} + \frac{V\alpha b(-x)}{\sqrt{1-x^2}} + \frac{\dot{h}b(-x)}{\sqrt{1-x^2}} + \frac{1}{2}\dot{\alpha}b^2\sqrt{1-x^2} \\ &\quad + \frac{\dot{\alpha}b^2(\frac{x}{2}-a)(-x)}{\sqrt{1-x^2}} = finite\,at\,x=1\end{aligned} \tag{24}$$

Therefore,

$$\left[\sqrt{1-x^2}\,\frac{\partial\varphi_\Gamma}{\partial x}\right]_{x=1} + \left\{-V\alpha b - \dot{h}b - \dot{\alpha}b^2(\frac{1}{2}-a)\right\} = 0 \tag{25}$$

Since

$$\frac{\partial\Delta\varphi_\Gamma}{\partial x} = \frac{\Delta\Gamma}{2\pi}\frac{\sqrt{x_0^{\,2}-1}}{\sqrt{1-x^2}}\frac{1}{x_0-x} \tag{26}$$

we can get the following expression from the above equation.

$$\begin{aligned}\left[\sqrt{1-x^2}\,\frac{\partial\varphi_\Gamma}{\partial x}\right]_{x=1} &= \int\frac{\Delta\Gamma}{2\pi}\frac{\sqrt{x_0^{\,2}-1}}{x_0-1} \\ &= \frac{b}{2\pi}\int_1^\infty\frac{\sqrt{x_0^{\,2}-1}}{x_0-1}\gamma dx_0 \\ &= V\alpha b + \dot{h}b + \dot{\alpha}b^2\left(\frac{1}{2}-a\right)\end{aligned} \tag{27}$$

Let's define

$$\frac{1}{2\pi}\int_1^\infty \frac{\sqrt{x_0{}^2-1}}{x_0-1}\gamma dx_0 = V\alpha + \dot{h} + \alpha b\left(\frac{1}{2}-a\right) \equiv Q \tag{28}$$

Then, the total force and moment on the airfoil will be

$$\begin{aligned} F_\Gamma &= -\rho V b\int_1^\infty \frac{x_0}{\sqrt{x_0{}^2-1}}\gamma dx_0 \\ &= -2\pi\rho V b Q\frac{\int_1^\infty \frac{x_0}{\sqrt{x_0{}^2-1}}\gamma dx_0}{\int_1^\infty \frac{\sqrt{x_0+1}}{\sqrt{x_0-1}}\gamma dx_0} \\ &= -2\pi\rho V b C Q \end{aligned} \tag{29}$$

$$\begin{aligned} M_\Gamma &= -\rho V b^2\int_1^\infty \left[\frac{1}{2}\sqrt{\frac{x_0+1}{x_0-1}} - \left(a+\frac{1}{2}\right)\frac{x_0}{\sqrt{x_0{}^2-1}}\right]\gamma dx_0 \\ &= -2\pi\rho V b^2 Q\left[\frac{1}{2} - C\left(a+\frac{1}{2}\right)\right] \end{aligned} \tag{30}$$

where C is the Theodorsen function, and defined as

$$C = \frac{\int_1^\infty \frac{x_0}{\sqrt{x_0{}^2-1}}\,\gamma\; dx_0}{\int_1^\infty \frac{\sqrt{x_0+1}}{\sqrt{x_0-1}}\,\gamma\; dx_0} \tag{31}$$

Let's assume that the airfoil has a simple harmonic motion

$$\gamma = \gamma_0 e^{i\left[k\left(\frac{s}{b}-x_0\right)+\varphi\right]} = \gamma_0 e^{i[\omega t - kx_0+\varphi]}$$

$$\text{where } s = Vt, k = \frac{\omega b}{V}, \frac{ks}{b} = \omega t$$

Then, Theodorsen function will be expressed as

$$
\begin{aligned}
C &= \frac{\int_1^\infty \frac{x_0}{\sqrt{{x_0}^2-1}}\,\gamma_0\, e^{i\omega t} e^{-ikx_0} e^{i\varphi}\, dx_0}{\int_1^\infty \frac{\sqrt{x_0+1}}{\sqrt{x_0-1}}\,\gamma_0\, e^{i\omega t} e^{-ikx_0} e^{i\varphi}\, dx_0} \\
&= \frac{\int_1^\infty \frac{x_0}{\sqrt{{x_0}^2-1}}\, e^{-ikx_0}\, dx_0}{\int_1^\infty \frac{\sqrt{x_0+1}}{\sqrt{x_0-1}}\, e^{-ikx_0}\, dx_0} \\
&= F(k) + iG(k)
\end{aligned}
\tag{32}
$$

The Theodorsen function can be expressed in terms of the following Bessel functions

$$
\begin{aligned}
C(k) &= \frac{-\frac{\pi}{2}J_1 + i\frac{\pi}{2}Y_1}{\left(-\frac{\pi}{2}J_1 - \frac{\pi}{2}Y_0\right) + \left(i\frac{\pi}{2}Y_1 - \frac{\pi}{2}J_0\right)} \\
&= \frac{-J_1 + iY_1}{-\left(J_1 + Y_0\right) + i\left(Y_1 - J_0\right)}
\end{aligned}
\tag{33}
$$

where J_1, Y_1 are the first and second kind Bessel functions. Also, the Theodorsen function can be expressed as

$$
C(k) = \frac{H_1^{(2)}(k)}{H_1^{(2)}(k) + iH_0^{(2)}(k)} \tag{34}
$$

where H_1, H_0 are the Hankel functions.

```
for i=1:5001,ri=(i-1)*0.001;yr(i)=theod(ri);xr(i)=ri;end
plot(xr,real(yr)),ylabel('F'),xlabel('k'),axis([0 5 0 1.2]),grid
plot(xr,imag(yr)),ylabel('G'),xlabel('k'),axis([0 5 -0.3 0]),grid

function thd = theod(rk)
% Theodorsen Function Calculation using modified bessel function
if(rk == 0), thd=1;
else
i=sqrt(-1);thd=BESSELK(1,i*rk)/(BESSELK(0,i*rk)+BESSELK(1,i*rk));
end
```

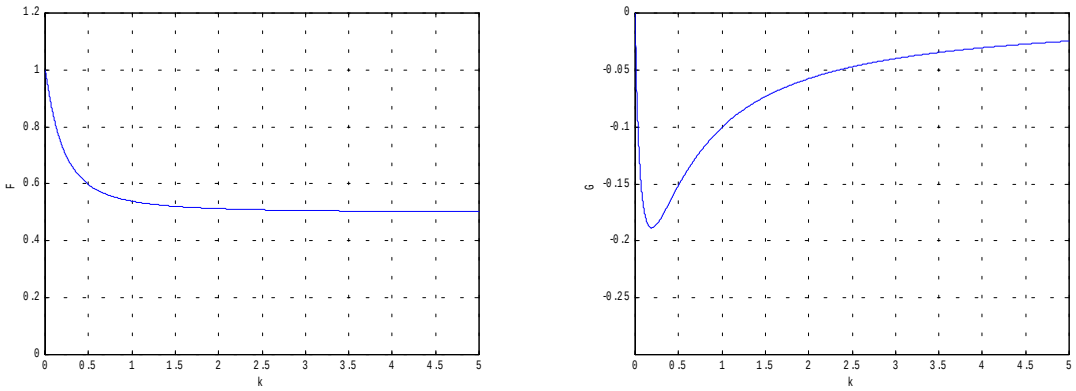


Figure Real and Imaginary part of the Theodorsen function

The Theodorsen function is frequentlyreplaced by simple algebraic approximations such as

$$C(k) = 0.5 + \frac{0.0075}{jk + 0.0455} + \frac{0.10055}{jk + 0.3}$$

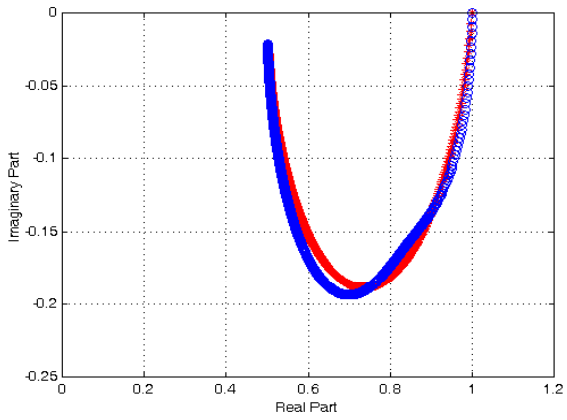


Figure Comparison of the Theodorsen function.

The total force and moment resulting from the noncirculatory and circulatory flows are expressed as

$$F = F_{NonCir} + F_{\Gamma} = -\pi\rho b^2 \left[\ddot{h} + V\dot{\alpha} - ba\ddot{\alpha}\right] - 2\pi\rho VbQC(k) \qquad (35)$$

$$M = \pi\rho b^2\left[ba\ddot{h} - Vb\left(\frac{1}{2}-a\right)\dot{\alpha} - b^2\left(\frac{1}{8}+a^2\right)\ddot{\alpha}\right] + 2\pi\rho Vb^2\left(a+\frac{1}{2}\right)QC(k) \qquad (36)$$

where

$$Q = V\alpha + \dot{h} + \dot{\alpha}b\left(\frac{1}{2}-a\right)$$

If we assume the quasi-steady aerodynamics (*The aerodynamic characteristics of an airfoil whose motion consists of variable linear and angular motions are equal, at any instant of time, to the characteristics of the same airfoil moving with constant linear and angular velocities equal to the actual instantaneous values.*), then *C(k)* becomes one, and the force and moment will be

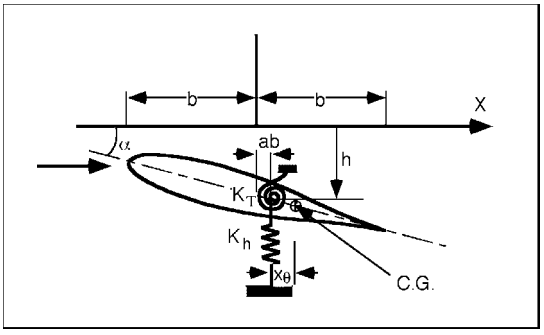
$$F_{QS} = -\pi\rho b^2\left[\ddot{h} + V\dot{\alpha} - ba\ddot{\alpha}\right] - 2\pi\rho Vb\left[V\alpha + \dot{h} + \dot{\alpha}b\left(\frac{1}{2}-a\right)\right] \qquad (37)$$

$$\begin{aligned} M_{QS} &= \pi\rho b^2\left[ba\ddot{h} - Vb\left(\frac{1}{2}-a\right)\dot{\alpha} - b^2\left(\frac{1}{8}+a^2\right)\ddot{\alpha}\right] \\ &\quad + 2\pi\rho Vb^2\left(a+\frac{1}{2}\right)\left[V\alpha + \dot{h} + \dot{\alpha}b\left(\frac{1}{2}-a\right)\right] \end{aligned} \qquad (38)$$

3.2. Flutter Equation of the Typical Section Model

3.2.a Equation of Motion

Consider the typical section shown in Figure.



The model has a translational spring with stiffness K_h and a torsion spring, with stiffness K_T. These springs are attached to the airfoil at the shear center. Therefore, it is two degrees of freedom model (h, α). And h is measured at the shear center (elastic axis).

The downward displacement of any other point on the airfoil is

$$z = h + x\alpha \tag{39}$$

where x is a distance measured from the shear center.

The strain energy and the kinetic energy are respectively given by

$$U = \frac{1}{2} K_T \alpha^2 + \frac{1}{2} K_h h^2 \tag{40}$$

$$T = \frac{1}{2} \int \rho \dot{z}^2 dx \tag{41}$$

where ρ be the mass per unit length of the airfoil.

$$T = \frac{1}{2} \left(\dot{h}^2 \int \rho dx + 2\dot{h}\dot{\alpha} \int \rho x dx + \dot{\alpha}^2 \int \rho x^2 dx \right) \tag{42}$$

Let's define the followings.

Mass $m = \int \rho dx$

The second moment of inertia of the airfoil about shear center, $I_\theta = \int \rho x^2 dx = m r_\theta^2$

The first moment of inertia of the airfoil about shear center, $S_\theta = \int \rho x dx = m x_\theta$

where r_θ is the radius of gyration and x_θ is a distance from the coordinate to the mass center.

Then, the kinetic energy can be rewritten as follows:

$$T = \frac{1}{2} m \dot{h}^2 + m x_\theta \dot{h}\dot{\alpha} + \frac{1}{2} I_\theta \dot{\alpha}^2 \tag{43}$$

The virtual work due to the unsteady aerodynamic forces is

$$\delta W_a = \int \Delta p \delta z dx = \int \Delta p \{\delta h + x\delta\alpha\} dx = Q_h \delta h + Q_\alpha \delta\alpha \tag{44}$$

where the force Q_h is positive downward, and moment Q_α is positive nose-up.

Lagrange's equations provide the equations of motion of the airfoil.

$$\frac{d}{dt}\left(\frac{\partial(T-U)}{\partial \dot{q}}\right) - \frac{\partial(T-U)}{\partial q} = Q_q \text{, where } q=h,\ \alpha \tag{45}$$

$$\begin{bmatrix} m & mx_\theta \\ mx_\theta & mr_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \end{Bmatrix} + \begin{bmatrix} K_h & 0 \\ 0 & K_T \end{bmatrix}\begin{Bmatrix} h \\ \alpha \end{Bmatrix} = \begin{Bmatrix} Q_h \\ Q_\theta \end{Bmatrix} = \begin{Bmatrix} F \\ M \end{Bmatrix}$$

$$\begin{bmatrix} 1 & \bar{x}_\theta \\ \bar{x}_\theta & \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} \ddot{h}/b \\ \ddot{\alpha} \end{Bmatrix} + \begin{bmatrix} \omega_h^2 & 0 \\ 0 & \omega_\theta^2 \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} = \begin{Bmatrix} F/mb \\ M/mb^2 \end{Bmatrix} \tag{46}$$

where $\omega_h^2 = K_h/m$, $\omega_\theta^2 = K_T/I_\theta$, $\bar{x}_\theta = x_\theta/b$, $\bar{r}_\theta = r_\theta/b$.

If we assume the harmonic motion $h = h_0 e^{i\omega t}$, $\alpha = \alpha_0 e^{i\omega t}$, the equation of motion will be

$$-\omega^2\begin{bmatrix} 1 & \bar{x}_\theta \\ \bar{x}_\theta & \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} + \begin{bmatrix} \omega_h^2 & 0 \\ 0 & \omega_\theta^2 \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} = \begin{Bmatrix} F/mb \\ M/mb^2 \end{Bmatrix} \tag{47}$$

The unsteady aerodynamic force and moment will be

$$\begin{aligned} F &= -\pi\rho b^2\left[\ddot{h} + V\dot{\alpha} - ba\ddot{\alpha}\right] - 2\pi\rho VbC(k)\left[V\alpha + \dot{h} + b\left(\frac{1}{2}-a\right)\dot{\alpha}\right] \\ &= \pi\rho b^3\omega^2\left[\frac{h}{b}\left(1 - i2C\frac{1}{k}\right) + \alpha\left(-a - i\frac{1}{k} - 2C\frac{1}{k^2} - i2\left(\frac{1}{2}-a\right)\frac{C}{k}\right)\right] \\ &= \pi\rho b^3\omega^2\left[\frac{h}{b}L_h + \alpha\left(L_\alpha - \left(\frac{1}{2}+a\right)L_h\right)\right] \end{aligned} \tag{48}$$

where the reduced frequency is $k = \frac{\omega b}{V}$, and $L_h = 1 - i2C\frac{1}{k}$, $L_\alpha = \frac{1}{2} - i\frac{1+2C}{k} - \frac{2C}{k^2}$.

Similarily,

$$M = \pi\rho b^4\omega^2\left[\left\{M_h - \left(\frac{1}{2}+a\right)L_h\right\}\frac{h}{b} + \left\{M_\alpha - \left(\frac{1}{2}+a\right)\left(L_\alpha + M_h\right) + \left(\frac{1}{2}+a\right)^2 L_h\right\}\alpha\right] \tag{49}$$

where $M_h = \frac{1}{2}, M_\alpha = \frac{3}{8} - i\frac{1}{k}$

Then, the equation motion can be rewritten as

$$\begin{aligned} &-\omega^2\begin{bmatrix} 1 & \bar{x}_\theta \\ \bar{x}_\theta & \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} + \begin{bmatrix} \omega_h^2 & 0 \\ 0 & \omega_\theta^2\bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \\ &= \frac{\omega^2}{\mu}\begin{bmatrix} L_h & L_\alpha - \left(\frac{1}{2}+a\right)L_h \\ M_h - \left(\frac{1}{2}+a\right)L_h & M_\alpha - \left(\frac{1}{2}+a\right)\left(L_\alpha + M_h\right) + \left(\frac{1}{2}+a\right)^2 L_h \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \end{aligned} \tag{50}$$

where the mass ration is defined as $\mu = \frac{m}{\pi\rho b^2 l}$. m is the airfoil mass per length, *l*.

Let's define $\Omega^2 = \frac{\omega^2}{\omega_\theta^2}$ and $R^2 = \frac{\omega_h^2}{\omega_\theta^2}$, then

$$\begin{aligned} &-\Omega^2\begin{bmatrix} 1 & \bar{x}_\theta \\ \bar{x}_\theta & \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} + \begin{bmatrix} R^2 & 0 \\ 0 & \bar{r}_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \\ &= \frac{\Omega^2}{\mu}\begin{bmatrix} L_h & L_\alpha - \left(\frac{1}{2}+a\right)L_h \\ M_h - \left(\frac{1}{2}+a\right)L_h & M_\alpha - \left(\frac{1}{2}+a\right)\left(L_\alpha + M_h\right) + \left(\frac{1}{2}+a\right)^2 L_h \end{bmatrix}\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \end{aligned} \tag{51}$$

3.2.b V-g method for flutter analysis

Let's express the above flutter equation in the following matrix form.

$$\left[K_{ij}\right]\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} = \Omega^2\left[A_{ij} + M_{ij}\right]\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \tag{52}$$

where K_{ij} is the stiffness matrix, M_{ij} mass matrix, and A_{ij} is the aerodynamic matrix. Note that the aerodynamic matrix is funtion of the reduced frequency, k.

V-g method assumes first the artifial structural damping, g.

$$[K_{ij}] = (1 + ig)[K_{ij}] \tag{53}$$

This artificial damping indicates the required damping for the harmonic motion. The eigenvalues of the equation of motion represent a point on the flutter boundary if the corresponding value of g equals the assumed value of g.

For give reduced frequency, $k = \dfrac{\omega b}{V}$, it is a complex eigenvalue problem.

$$\frac{(1+ig)}{\Omega^2}[K_{ij}]\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} = [A_{ij} + M_{ij}]\begin{Bmatrix} h/b \\ \alpha \end{Bmatrix} \tag{54}$$

The eigenvalue is

$$\lambda = \frac{1+\mathrm{ig}}{\Omega^2} \tag{55}$$

From this eigenvalue we have

$$\frac{1}{\lambda_{\mathrm{Re}}} = \frac{\omega_{\mathrm{i}}^2}{\omega_\theta^2} \tag{56}$$

$$\mathrm{g} = \frac{\lambda_{\mathrm{Im}}}{\lambda_{\mathrm{Re}}} \tag{57}$$

The complex eigenvalue problem is solved beginning with large values of k and then decreasing k until a flutter velocity is found. If there is no actual damping in the system, when the artificial damping, g, first becomes positive, flutter will occur.

Example

Calculate the flutter speed of the following typical section model.

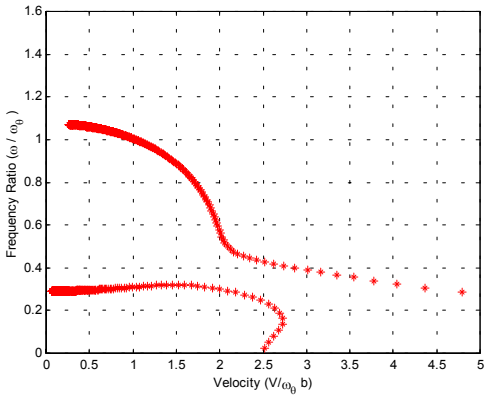
x_θ=0.2, r_θ=0.5, μ=20, a=-0.1, b=1, and R=0.3

```
% 2 dof system
% xth ; x _ theta
```

```
% rth2 ; ( r _theta)^2
% R2 ; (omega_h /omega alpha)^2
xth =0.2;     R2  =(0.3)^2;
rth2=(0.5)^2; mu  =20.0 ;   a   =-0.1 ;
%
i=sqrt(-1);
result1=zeros(2,400); result2=zeros(2,400); veloci=zeros(2,400);
for kk=400:-1:1;    rk=kk*0.01;
%
% Theodorsen Function Calculation
C=0.5+0.0075/(i*rk+0.0455)+0.10055/(i*rk+0.3);
% Mass
ms=[ 1     xth
     xth   rth2];
%
% stiffness matrix
ks=[ R2  0
     0   rth2];
%
lh=1-i*2*C/rk;    la=.5-i*(1+2*C)/rk-2*C/rk/rk;
mh=0.5;           ma=3/8-i/rk;
aero=[ lh              la-(.5-a)*lh
       mh-(.5+a)*lh    ma-(.5+a)*(la+mh)+(.5+a)*(.5+a)*lh]/mu;

ddd=eig(inv(ks)*(aero+ms));  rrr=real(ddd); iii= imag(ddd);
rst1(:,kk)=sqrt(1./rrr);  rst2(:,kk)=iii./rrr;
vel(:,kk)=sqrt(1./rrr)/rk;
end

figure(1);
plot(vel(1,:),rst1(1,:),'.r',vel(2,:),rst1(2,:),'.r'),
axis([0.0 5.00 0 1.60]),xlabel('Velocity'),ylabel('Frequency Ratio'),grid;
figure(2);
plot(vel(1,:),rst2(1,:),'.r',vel(2,:),rst2(2,:),'.r'),
axis([0.0 5.00 -1.2 1.0]),xlabel('Velocity'),ylabel('g'),grid;
```



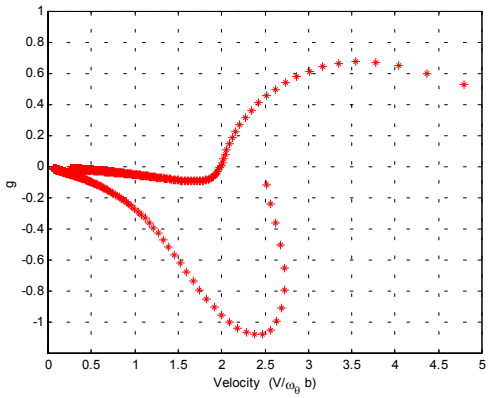


Figure Plots of $\frac{\omega_i}{\omega_\theta}$ and g versus $\frac{V}{\omega_\theta b}$

Flutter speed of the model is $\frac{V_{Flutter}}{\omega_\theta b} = 1.99$, frequency ration $\frac{\omega_{Flutter}}{\omega_\theta} = 0.58$. From V-g plot results, Can you estimate the divergence speed of this airfoil ?

3.3. Typical Section with the Trailing Edge Flap

Let's put the trailing edge flap at the airfoil that we discussed in the previous section. The trailing edge flap is hinged at x=bc. Again the airfoil has a translational spring as well as the torsional spring at the shear center. The same way that we discussed at the previous section, we can calculate the unsteady aerodynamic forces.

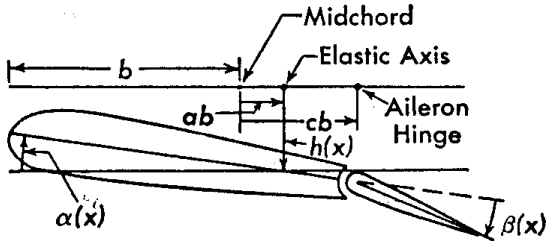


Figure Typical Section with the Trailing Edge Control Surface

The displacement of the airfoil at x will be

$$z = h + \alpha(x - ab) + \beta\,(x - bc)U(x - bc) \tag{58}$$

where x is the distance measured from the midchord and *U(x-bc)* is the unit step function. Then, the upwash will be

$$\begin{aligned} w_a(x,t) &= -\left(\frac{\partial z}{\partial t} + V\frac{\partial z}{\partial x}\right) \\ &= -[\dot{h} + \dot{\alpha}(x - ab)] - V\alpha - \dot{\beta}(x - bc)U(x - bc) - V\beta U(x - bc) \end{aligned} \tag{59}$$

The total force and moment resulting from the noncirculatory and circulatory flows are expressed as

$$F = -\pi\rho b^2\left[\ddot{h} + V\dot{\alpha} - ba\ddot{\alpha} - \frac{V}{\pi}T_4\dot{\beta} - \frac{b}{\pi}T_1\ddot{\beta}\right] - 2\pi\rho VbQC(k) \tag{60}$$

$$\begin{aligned} M = {} & \pi\rho b^2\left[ba\ddot{h} - Vb\left(\frac{1}{2} - a\right)\dot{\alpha} - b^2\left(\frac{1}{8} + a^2\right)\ddot{\alpha} - \frac{V^2}{\pi}(T_4 + T_{10})\beta\right. \\ & \left. + \frac{Vb}{\pi}\left\{-T_1 + T_8 + (c-a)T_4 - \frac{1}{2}T_{11}\right\}\dot{\beta} + \frac{b^2}{\pi}\{T_7 + (c-a)T_1\}\ddot{\beta}\right] \\ & + 2\pi\rho Vb^2\left(a + \frac{1}{2}\right)QC(k) \end{aligned} \tag{61}$$

$$\begin{aligned} M_\beta = {} & \pi\rho b^2\left[\frac{b}{\pi}T_1\ddot{h} + \frac{Vb}{\pi}\left\{2T_9 + T_1 - (a - \frac{1}{2})T_4\right\}\dot{\alpha}\right. \\ & - \frac{2b^2}{\pi}T_{13}\ddot{\alpha} - \left(\frac{V}{\pi}\right)^2(T_5 - T_4T_{10})\beta \\ & \left. + \frac{Vb}{2\pi^2}T_4T_{11}\dot{\beta} + \left(\frac{b}{\pi}\right)^2 T_3\ddot{\beta}\right] - \rho Vb^2T_{12}QC(k) \end{aligned} \tag{62}$$

where

$$Q = V\alpha + \dot{h} + \dot{\alpha}b\left(\frac{1}{2} - a\right) + \frac{V}{\pi}T_{10}\beta + \frac{b}{2\pi}T_{11}\dot{\beta} \tag{63}$$

T functions are defined as follows.

$$T_1 = -\frac{2 + c^2}{3}\sqrt{1 - c^2} + c\cos^{-1}c$$

$$T_3 = -\frac{1\text{-}c^2}{8}(5c^2 + 4) + \frac{1}{4}c(7 + 2c^2)\sqrt{1 - c^2}\cos^{-1}c - (\frac{1}{8} + c^2)(\cos^{-1}c)^2$$

$$T_4 = c\sqrt{1 - c^2} - \cos^{-1}c$$

$$T_5 = -(1\text{-}c^2) - (\cos^{-1}c)^2 + 2c\sqrt{1 - c^2}\cos^{-1}c$$

$$T_7 = c\frac{7 + 2c^2}{8}\sqrt{1 - c^2} - (\frac{1}{8} + c^2)\cos^{-1}c$$

$$T_8 = -\frac{1}{3}(1 + 2c^2)\sqrt{1 - c^2} + c\cos^{-1}c$$

$$T_9 = \frac{1}{2}\left[\frac{\sqrt{1-c^2}(1\text{-}c^2)}{3} + aT_4\right]$$

$$T_{10} = \sqrt{1-c^2} + \cos^{-1} c$$

$$T_{11} = (2\text{-}c)\sqrt{1-c^2} + (1\text{-}2c)\cos^{-1} c$$

$$T_{12} = (2+c)\sqrt{1-c^2}\text{-}(1+2c)\cos^{-1} c$$

$$T_{13} = -\frac{1}{2}\left[T_7 + (c\text{-}a)T_1\right]$$

$$T_{15} = T_4 + T_{10}$$

$$T_{16} = T_1\text{-}T_8\text{-}(c\text{-}a)T_4 + \frac{1}{2}T_{11}$$

$$T_{17} = -2T_9\text{-}T_1 + (a\text{-}\frac{1}{2})T_4$$

$$T_{18} = T_5\text{-}T_4T_{10}$$

$$T_{19} = -\frac{1}{2}T_4T_{11}$$

Note that Q can be rewritten as the following form by using S_1 and S_2

$$Q = V\lfloor S_1 \rfloor\{x_s\} + b\lfloor S_2 \rfloor\{\dot{x}_s\} \tag{64}$$

where

$$\mathrm{S}_1 = \left[0 \quad 1 \quad \frac{T_{10}}{\pi}\right]$$

$$\mathrm{S}_2 = \left[1 \quad (0.5-a) \quad \frac{T_{11}}{2\pi}\right]$$

$$\mathrm{x}_s = \begin{Bmatrix} h/b \\ \alpha \\ \beta \end{Bmatrix}$$

In matrix form, the aerodynamic forces and moments will be

$$\begin{Bmatrix} F/b \\ M/b^2 \\ M_\beta/b^2 \end{Bmatrix} = q\Bigg[2C(k)\{R\}\lfloor S_1 \rfloor \{x_s\} + \frac{2b}{V} C(k)\{R\}\lfloor S_2 \rfloor \{\dot{x}_s\} + \frac{2b^2}{V^2}[M_{nc}]\{\ddot{x}_s\} + \frac{2b}{V}[B_{nc}]\{\dot{x}_s\} + 2[K_{nc}]\{x_s\} \Bigg] \tag{65}$$

where

$$\mathrm{R} = \begin{Bmatrix} -2\pi \\ 2\pi(a+0.5) \\ -T_{12} \end{Bmatrix}$$

The subscript nc indicates the non-circulatory part.

$$M_{nc} = \begin{bmatrix} -\pi & \pi a & T_1 \\ \pi a & -\pi(\frac{1}{8}+a^2) & -2T_{13} \\ T_1 & -2T_{13} & \frac{T_3}{\pi} \end{bmatrix}$$

$$B_{nc} = \begin{bmatrix} 0 & -\pi & T_4 \\ 0 & \pi(a-\frac{1}{2}) & -T_{16} \\ 0 & -T_{17} & -\frac{T_{19}}{\pi} \end{bmatrix}$$

$$K_{nc} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_{15} \\ 0 & 0 & -\frac{T_{18}}{\pi} \end{bmatrix}$$

The equation of motion of the airfoil are derived from Lagrange's equations.

$$\frac{d}{dt}\left(\frac{\partial(T-U)}{\partial \dot{q}}\right) - \frac{\partial(T-U)}{\partial q} = Q_q \text{, where } q=h,\ \alpha,\ \beta \tag{66}$$

The kinetic energy is

$$T = \frac{1}{2}\int \dot{z}^2 \rho dx \tag{67}$$

where airfoil deflection is

$$z = -h - x\alpha - \big(x - (bc - ba)\big)\beta U\big(x - (bc - ba)\big) \tag{68}$$

It has to be noted that the coordinate is placed at the shear center.

$$T = \frac{1}{2}m\dot{h}^2 + mx_\theta \dot{h}\dot{\alpha} + \frac{1}{2}I_\theta \dot{\alpha}^2 + \frac{1}{2}I_\beta \dot{\beta}^2 + mx_\beta \dot{h}\dot{\beta} + \big(I_\beta + mx_\beta (bc - ba)\big)\dot{\alpha}\dot{\beta} \tag{69}$$

The potential energy U is stored in springs attached at the control surface hinge lines and at the elastic axis.

$$U = \frac{1}{2}(K_h h^2 + K_T \alpha^2 + K_\beta \beta^2) \tag{70}$$

$$\begin{bmatrix} m & mx_\theta & mx_\beta \\ mx_\theta & mr_\theta^{\,2} & mr_\theta^{\,2} + mx_\beta (bc - ba) \\ mx_\beta & mr_\theta^{\,2} + mx_\beta (bc - ba) & mr_\beta^{\,2} \end{bmatrix} \begin{Bmatrix} \ddot{h} \\ \ddot{\alpha} \\ \ddot{\beta} \end{Bmatrix} + \begin{bmatrix} K_h & 0 & 0 \\ 0 & K_T & 0 \\ 0 & 0 & K_\beta \end{bmatrix} \begin{Bmatrix} h \\ \alpha \\ \beta \end{Bmatrix} = \begin{Bmatrix} F \\ M \\ M_\beta \end{Bmatrix} \tag{71}$$

$$
\begin{bmatrix} 1 & \bar{x}_\theta & \bar{x}_\beta \\ \bar{x}_\theta & \bar{r}_\theta^{\,2} & \bar{r}_\theta^{\,2} + \bar{x}_\beta(c-a) \\ \bar{x}_\beta & \bar{r}_\theta^{\,2} + \bar{x}_\beta(c-a) & \bar{r}_\beta^{\,2} \end{bmatrix} \begin{Bmatrix} \ddot{h}/b \\ \ddot{\alpha} \\ \ddot{\beta} \end{Bmatrix} + \begin{bmatrix} \omega_h^2 & 0 & 0 \\ 0 & \omega_\theta^2 \bar{r}_\theta^{\,2} & 0 \\ 0 & 0 & \omega_\beta^2 \bar{r}_\beta^{\,2} \end{bmatrix} \begin{Bmatrix} h/b \\ \alpha \\ \beta \end{Bmatrix}
$$

$$
= \frac{q}{m} \Bigg[2C(k) \left\{ \begin{Bmatrix} -2\pi \\ 2\pi(a+0.5) \\ -T_{12} \end{Bmatrix} \begin{bmatrix} 0 & 1 & \dfrac{T_{10}}{\pi} \end{bmatrix} \begin{Bmatrix} h/b \\ \alpha \\ \beta \end{Bmatrix} \right.
$$

$$
\left. + \frac{2b}{V} \begin{Bmatrix} -2\pi \\ 2\pi(a+0.5) \\ -T_{12} \end{Bmatrix} \begin{bmatrix} 1 & (0.5-a) & \dfrac{T_{11}}{2\pi} \end{bmatrix} \begin{Bmatrix} \dot{h}/b \\ \dot{\alpha} \\ \dot{\beta} \end{Bmatrix} \right\}
$$

$$
+ \frac{2b^2}{V^2} \begin{bmatrix} -\pi & \pi a & T_1 \\ \pi a & -\pi\left(\dfrac{1}{8} + a^2\right) & -2T_{13} \\ T_1 & -2T_{13} & \dfrac{T_3}{\pi} \end{bmatrix} \begin{Bmatrix} \ddot{h}/b \\ \ddot{\alpha} \\ \ddot{\beta} \end{Bmatrix}
$$

$$
+ \frac{2b}{V} \begin{bmatrix} 0 & -\pi & T_4 \\ 0 & \pi\left(a - \dfrac{1}{2}\right) & -T_{16} \\ 0 & -T_{17} & -\dfrac{T_{19}}{\pi} \end{bmatrix} \begin{Bmatrix} \dot{h}/b \\ \dot{\alpha} \\ \dot{\beta} \end{Bmatrix} \qquad (72)
$$

$$
+ 2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_{15} \\ 0 & 0 & -\dfrac{T_{18}}{\pi} \end{bmatrix} \begin{Bmatrix} h/b \\ \alpha \\ \beta \end{Bmatrix} \Bigg]
$$

Or

$$[\overline{M}]\{\ddot{x}_s\}+[\overline{K}]\{x_s\}=\frac{q}{m}\Bigg(2C(k)\{R\}\lfloor S_1\rfloor\{x_s\}+\frac{2b}{V}C(k)\{R\}\lfloor S_2\rfloor\{\dot{x}_s\} + \frac{2b^2}{V^2}[M_{nc}]\{\ddot{x}_s\}+\frac{2b}{V}[B_{nc}]\{\dot{x}_s\}+2[K_{nc}]\{x_s\}\Bigg) \tag{73}$$

V-g method for flutter analysis

Let's assume the harmonic motion of the system, $x_s = x_{s0}e^{i\omega t}$

The equation of motion will be expressed in the following matrix form.

$$-\omega^2[\overline{M}]\{x_s\}+[\overline{K}]\{x_s\}=\frac{q}{m}\Bigg(2C(k)\{R\}\lfloor S_1\rfloor\{x_s\}+(i\omega)\frac{2b}{V}C(k)\{R\}\lfloor S_2\rfloor\{x_s\} + (-\omega^2)\frac{2b^2}{V^2}[M_{nc}]\{x_s\}+(i\omega)\frac{2b}{V}[B_{nc}]\{x_s\}+2[K_{nc}]\{x_s\}\Bigg) \tag{74}$$

$$\frac{1}{\omega^2}[\overline{K}]\{x_s\}=[\overline{M}]\{x_s\}+\frac{1}{\mu}\Bigg(\frac{1}{k^2}C(k)\{R\}\lfloor S_1\rfloor\{x_s\}+i\frac{1}{k}C(k)\{R\}\lfloor S_2\rfloor\{x_s\} -[M_{nc}]\{x_s\}+i\frac{1}{k}[B_{nc}]\{x_s\}+\frac{1}{k^2}[K_{nc}]\{x_s\}\Bigg) \tag{75}$$

where μ is the mass ratio, k is the reduced frequency.

In V-g method, the artifial structural damping, g is assumed.

$$\frac{1+ig}{\omega^2}[\overline{K}]\{x_s\}=[\overline{M}]\{x_s\}+\frac{1}{\mu}\Bigg(\frac{1}{k^2}C(k)\{R\}\lfloor S_1\rfloor\{x_s\}+i\frac{1}{k}C(k)\{R\}\lfloor S_2\rfloor\{x_s\} -[M_{nc}]\{x_s\}+i\frac{1}{k}[B_{nc}]\{x_s\}+\frac{1}{k^2}[K_{nc}]\{x_s\}\Bigg) \tag{76}$$

Again, it's a complex eigenvalue problem. For give reduced frequency, $k=\frac{\omega b}{V}$,

the eigenvalue, $\lambda=\frac{1+\mathrm{ig}}{\omega^2}$ is obtained, and we will have

$$\frac{1}{\lambda_{\mathrm{Re}}}=\omega_{\mathrm{i}}^2$$

$$g = \frac{\lambda_{\mathrm{Im}}}{\lambda_{\mathrm{Re}}}$$

```
% 3 dof system
wh=50.0;   wa=100.0;    wb=300.0;
a=-0.4;    c=0.6;       b=1;
xa=0.2;    xb=0.0125;
ra=sqrt(0.25);          rb=sqrt(0.00625);
mu=40;

nn=3;i=sqrt(-1);rho=0.002378;
mas=pi*rho*mu*b*b;  damb=0.0;
sqr=sqrt(1-c*c);
arc=acos(c);
%
%
% final version of t function generating the aero matrix
t1=-(2+c*c)/3*sqr+c*arc;
t3=-(1-c*c)/8*(5.*c*c+4)+.25*c*(7+2*c*c)*sqr*arc-(1./8.+c*c)*arc*arc;
t4=c*sqr-arc;
t5=-(1-c*c)-arc*arc+2*c*sqr*arc;
t7=c*(7+2*c*c)/8*sqr-(1/8+c*c)*arc;
t8=-1/3*(1+2*c*c)*sqr+c*arc;
t9=.5*(sqr*(1-c*c)/3+a*t4);
t10=sqr+arc;
t11=(2-c)*sqr+(1-2*c)*arc;
t12=(2+c)*sqr-(1+2*c)*arc;
t13=-.5*(t7+(c-a)*t1);
t15=t4+t10;
t16=t1-t8-(c-a)*t4+0.5*t11;
t17=-2*t9-t1+(a-.5)*t4;
t18=t5-t4*t10;
t19=-.5*t4*t11;
%
iden=zeros(nn,nn);ks=zeros(nn,nn);knc=zeros(nn,nn);bs=zeros(nn,nn);
   for ii=1:nn
   iden(ii,ii)=1.0;
   end
% mass matrix
ms(1,1)=1;    ms(1,2)=xa;              ms(1,3)=xb;
ms(2,1)=xa;   ms(2,2)=ra*ra;           ms(2,3)=rb*rb+xb*(c-a);
ms(3,1)=xb;   ms(3,2)=rb*rb+xb*(c-a);  ms(3,3)=rb*rb;
%
%
mnc(1,1)=-pi;  mnc(1,2)=pi*a;             mnc(1,3)=t1;
mnc(2,1)=pi*a; mnc(2,2)=-pi*(1./8.+a*a);  mnc(2,3)=-2*t13;
mnc(3,1)=t1;   mnc(3,2)=-2*t13;           mnc(3,3)=t3/pi;
```

```
%
%
% stiffness matrix
ks(1,1)=wh*wh;
ks(2,2)=ra*ra*wa*wa;
ks(3,3)=rb*rb*wb*wb;
%
%
knc(2,3)=-t15;
knc(3,3)=-t18/pi;
%
%
%
bnc(1,1)=0;    bnc(1,2)=-pi;        bnc(1,3)=t4;
bnc(2,1)=0;    bnc(2,2)=pi*(a-.5);  bnc(2,3)=-t16;
bnc(3,1)=0;    bnc(3,2)=-t17;       bnc(3,3)=-t19/pi;
%
%
%
r1=[    -2.*pi
     2*pi*(a+0.5)
        -t12];
%
s1=[0  1    t10/pi];
s2=[1 (.5-a)  t11/(2*pi)];
%
% v-g method

m=200;
rst1=zeros(3,m); rst2=zeros(3,m); vel=zeros(3,m);
    for kk=m:-1:1;
    rk=kk*0.02;
   [f,g]=faero(rk,b,mnc,bnc,knc,r1,s1,s2,mu);
   aero=f+i*g;
   ddd=eig(inv(ks)*(ms+aero));
   rrr=abs(real(ddd)); iii= imag(ddd);
   rst1(:,kk)=sqrt(1./rrr);  rst2(:,kk)=iii./rrr;
   vel(:,kk)=sqrt(1./rrr)*b/rk;
   end

xxx=[vel(1,:);  rst1(1,:);  rst2(1,:);  vel(2,:);  rst1(2,:);  rst2(2,:);
vel(3,:); rst1(3,:); rst2(3,:)]';
figure(1);
plot(vel(1,:),rst1(1,:),'*r',vel(2,:),rst1(2,:),'*r',vel(3,:),rst1(3,:)
,'*r'),
axis([0.0 500 0 500]),xlabel('Velocity'),ylabel('Frequency'),grid;
figure(2);
plot(vel(1,:),rst2(1,:),'*r',vel(2,:),rst2(2,:),'*r',vel(3,:),rst2(3,:)
,'*r'),
axis([0.0 500 -0.5 0.5]),xlabel('Velocity'),ylabel('g'),grid;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file : faero.m
% aero forces in frequency domain
function [f,g]=faero(rk,b,mnc,bnc,knc,r1,s1,s2,mu)
i=sqrt(-1);cs=theod(rk);
z=-mnc+i*(bnc+cs*r1*s2)/rk+(knc+cs*r1*s1)/(rk^2);
z=z/(mu*pi);f=real(z);g=imag(z);
%
```

Example

Calculate the flutter speed of the following typical section model.

$\omega_h = 50.0\ rad/\sec,\ \omega_\theta = 100.0\ rad/\sec,\ \omega_\beta = 300.0 rad/\sec$

$a = \text{-}0.4,\ c = 0.6, b = 1, \bar{x}_\theta = 0.2,\ \bar{x}_\beta = 0.0125, \bar{r}_\theta^2 = 0.25,\ \bar{r}_\beta^2 = 0.00625, \mu = 40$

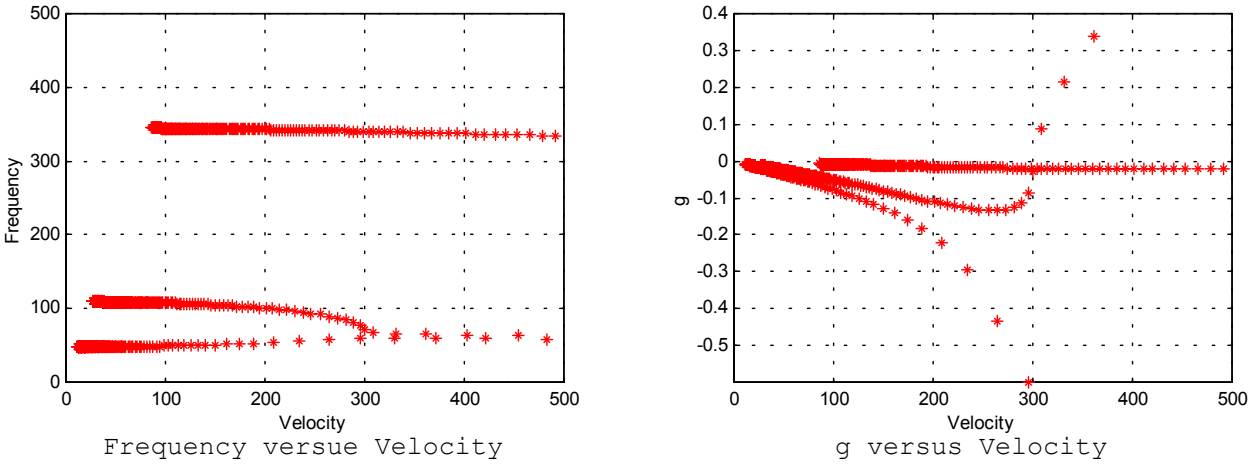


Figure . Flutter Result

3.4. Rational Function Approximation of the Unsteady Aerodynamic Forces

For the aeroservoelastic analysis and design, it is necessary to transform the equations of motion into the state space form. This requires to approximate the frequency domain unsteady aerodynamic forces in terms of rational functions of the Laplace variable. There are several methods for the rational function approximation (RFA) such as Roger's method, Matrix Pade method, and Minimum State method. However, when the RFA is

conducted, it causes an increase in the total number of states due to the number of augmented aerodynamic states to represent unsteady aerodynamic forces accurately.

In this section, we will discuss about Roger's method since Roger's method is very simple and accurately transforms the unsteady aerodynamic forces from frequency domain into time domain. The minimum state method will be discussed in the next chapter,

First, let's express the unsteady aerodynamic forces in the frequency domain as follows;

$$\begin{Bmatrix} F/b \\ M/b^2 \\ M_\beta/b^2 \end{Bmatrix} = q\Bigg[2C(k)\{R\}\lfloor S_1 \rfloor \{x_s\} + \frac{2b}{V} C(k)\{R\}\lfloor S_2 \rfloor \{\dot{x}_s\} + \frac{2b^2}{V^2}[M_{nc}]\{\ddot{x}_s\} + \frac{2b}{V}[B_{nc}]\{\dot{x}_s\} + 2[K_{nc}]\{x_s\} \Bigg]$$
$$= q\Big[2C(k)\{R\}\lfloor S_1 \rfloor + i2kC(k)\{R\}\lfloor S_2 \rfloor - 2k^2[M_{nc}] + i2k[B_{nc}] + 2[K_{nc}]\Big]\{x_s\} \quad (77)$$
$$= q\Big[2C(k)\{R\}\lfloor S_1 \rfloor + 2s'C(k)\{R\}\lfloor S_2 \rfloor + 2s'^2[M_{nc}] + 2s'[B_{nc}] + 2[K_{nc}]\Big]\{x_s\}$$
$$= qA(s')\{x_s\}$$

where *s'* is the nondimensionalized Laplace variable $s' = ik = sb/V$

Roger's method approximates the unsteady aerodynamic forces in the following form.

$$[A_{ap}] = [P_0] + [P_1]s' + [P_2]s'^2 + \sum_{j=3}^{N} \frac{\lfloor P_j \rfloor s'}{s' + \gamma_{j-2}} \quad (78)$$

γ_{j-2} is the aerodynamic poles which are usually preselected in the range of reduced frequencies of interest.

Let's define the calculated aerodynamic forces as $[A(s')] = [F(s')] + i[G(s')]$.

The real and imaginary part of the approximated aerodynamic matrix will be

$$\text{Real part of } \left[A_{ap}\right] = \left[P_0\right] + \left[P_2\right]s'^2 + \sum_{j=3}^{N} \frac{\left[P_j\right](-s'^2)}{(-s'^2) + \gamma_{j-2}^2} \tag{79}$$

$$\text{Imaginary part of } \left[A_{ap}\right] = \left[P_1\right]s' + \sum_{j=3}^{N} \frac{\left[P_j\right]\gamma_{j-2}s'}{(-s'^2) + \gamma_{j-2}^2} \tag{80}$$

Note that the nondimensionalized Laplace variable is *s'=ik*.

The real matrices $[P_j]$ are determined using least square technique for a term by term fitting of the aerodynamic matrix, $[A(s')]$.

$$\sum_i [Aaero]^T [Aaero]\{Xp\} = \sum_i [Aaero]^T \{Baero\} \tag{81}$$

where

$$\left[Aaero\right] = \begin{bmatrix} 1 & 0 & -k_i^2 & \cdots & \dfrac{k_i^2}{k_i^2 + \gamma_{j-2}^2} \\ 0 & k_i & 0 & \cdots & \dfrac{\gamma_{j-2}k_i}{k_i^2 + \gamma_{j-2}^2} \end{bmatrix}$$

$$\left\{Baero\right\} = \begin{Bmatrix} F(k_i) \\ G(k_i) \end{Bmatrix}_{mn}$$

$$\left\{Xp\right\} = \begin{Bmatrix} P_0 \\ P_1 \\ P_2 \\ \vdots \\ P_N \end{Bmatrix}_{mn}$$

The augmented aerodynamic state is defined as follows.

$$\{x_{aj}\} = \frac{s'}{s' + \gamma_{j-2}}\{x_s\} = \frac{s}{s + \dfrac{V}{b}\gamma_{j-2}}\{x_s\} \tag{82}$$

$$\{\dot{x}_{aj}\} = \{\dot{x}_s\} - \frac{V}{b}\gamma_{j-2}\{x_{aj}\} \tag{83}$$

Then, the state space equation of motion is expressed in the following form.

$$\begin{Bmatrix} \dot{x}_s \\ \ddot{x}_s \\ \dot{x}_{a3} \\ .. \\ \dot{x}_{aN} \end{Bmatrix} = \begin{bmatrix} 0 & I & 0 & .. & 0 \\ -\overline{M}^{-1}\overline{K} & -\overline{M}^{-1}\overline{B} & q\overline{M}^{-1}P_3 & .. & q\overline{M}^{-1}P_N \\ 0 & I & -\left(\frac{V}{b}\right)\gamma_1 I & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & I & 0 & 0 & -\left(\frac{V}{b}\right)\gamma_{N-2} I \end{bmatrix} \begin{Bmatrix} x_s \\ \dot{x}_s \\ x_{a3} \\ .. \\ x_{aN} \end{Bmatrix} \tag{84}$$

where

$$\overline{M} = M - qP_2\left(\frac{b}{V}\right)^2$$

$$\overline{B} = B - qP_1\left(\frac{b}{V}\right)$$

$$\overline{K} = K - qP_0$$

Example

Using the state space equation, calculate the flutter speed of the following typical section model. Assume four aerodynamic poles for the rational function approximation using Roger's method. (γ=0.2, 0.4, 0.6, 0.8)

$\omega_h = 50.0\ rad/\text{sec},\ \ \omega_\theta = 100.0\ rad/\text{sec},\ \ \omega_\beta = 300.0 rad/\text{sec}$

$a = -0.4,\ c = 0.6, b = 1, \bar{x}_\theta = 0.2,\ \bar{x}_\beta = 0.0125, \bar{r}_\theta^2 = 0.25,\ \bar{r}_\beta^2 = 0.00625, \mu = 40$

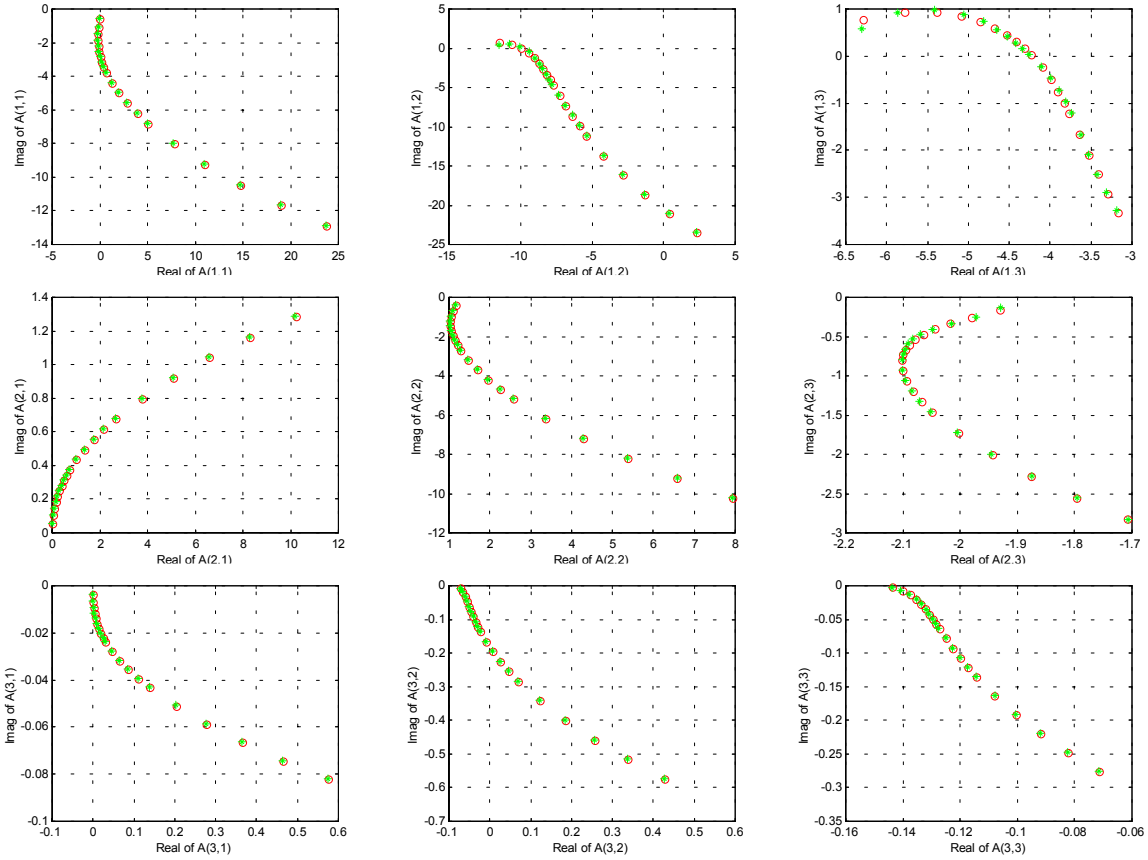
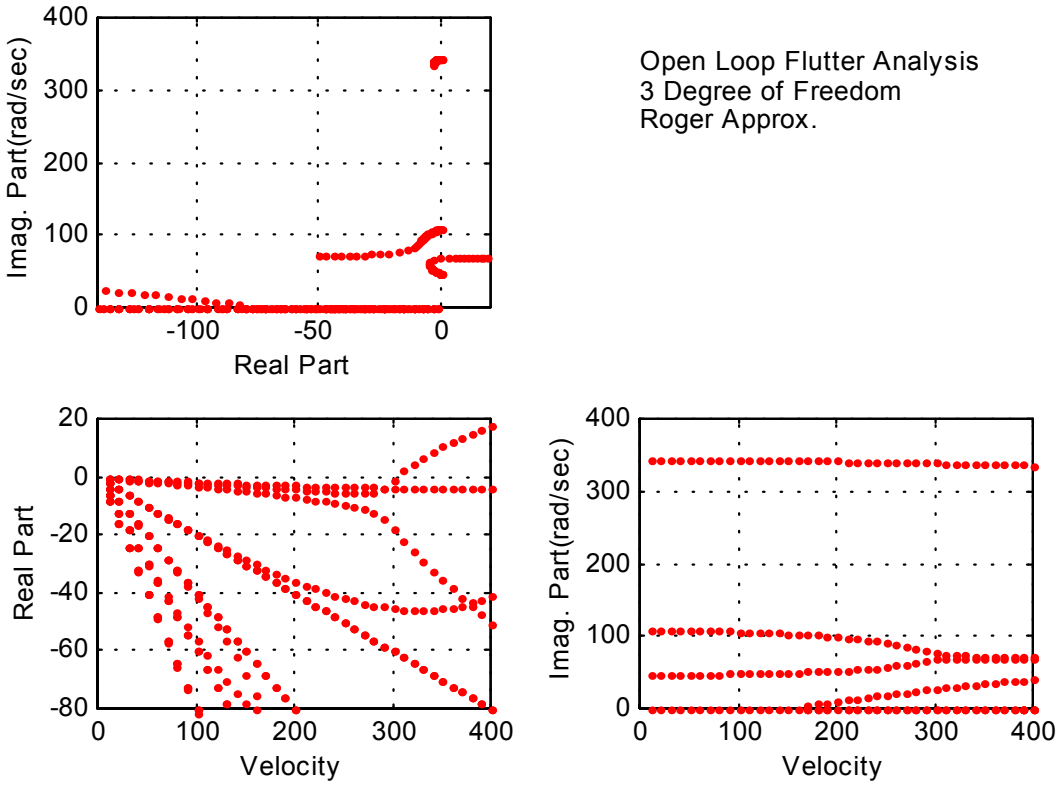


Figure Rational function approximation results



The result shows that the model has a flutter speed of about 300 ft/sec, and the flutter frequency is around 78 rad/sec, which is the pitching mode dominant.

```
% function [result]= roger3(wh,wa,wb,a,c,xa,xb)
% version 1992 1 6
% 3 dof system
wh=50.0;   wa=100.0;    wb=300.0;
a=-0.4;    c=0.6;       b=1;
xa=0.2;    xb=0.0125;
ra=sqrt(0.25);          rb=sqrt(0.00625);
mu=40;

nn=3;i=sqrt(-1);rho=0.002378;
mas=pi*rho*mu*b*b;  damb=0.0;
sqr=sqrt(1-c*c);
arc=acos(c);
%
%
% final version of t function generating the aero matrix
t1=-(2+c*c)/3*sqr+c*arc;
t3=-(1-c*c)/8*(5.*c*c+4)+.25*c*(7+2*c*c)*sqr*arc-(1./8.+c*c)*arc*arc;
t4=c*sqr-arc;
t5=-(1-c*c)-arc*arc+2*c*sqr*arc;
t7=c*(7+2*c*c)/8*sqr-(1/8+c*c)*arc;
t8=-1/3*(1+2*c*c)*sqr+c*arc;
t9=.5*(sqr*(1-c*c)/3+a*t4);
t10=sqr+arc;
t11=(2-c)*sqr+(1-2*c)*arc;
t12=(2+c)*sqr-(1+2*c)*arc;
t13=-.5*(t7+(c-a)*t1);
t15=t4+t10;
t16=t1-t8-(c-a)*t4+0.5*t11;
t17=-2*t9-t1+(a-.5)*t4;
t18=t5-t4*t10;
t19=-.5*t4*t11;
%
iden=zeros(nn,nn);ks=zeros(nn,nn);knc=zeros(nn,nn);bs=zeros(nn,nn);
   for ii=1:nn
   iden(ii,ii)=1.0;
   end
% mass matrix
ms(1,1)=1;    ms(1,2)=xa;              ms(1,3)=xb;
ms(2,1)=xa;   ms(2,2)=ra*ra;           ms(2,3)=rb*rb+xb*(c-a);
ms(3,1)=xb;   ms(3,2)=rb*rb+xb*(c-a);  ms(3,3)=rb*rb;
ms=ms*mas
%
%
mnc(1,1)=-pi;  mnc(1,2)=pi*a;             mnc(1,3)=t1;
mnc(2,1)=pi*a; mnc(2,2)=-pi*(1./8.+a*a);  mnc(2,3)=-2*t13;
mnc(3,1)=t1;   mnc(3,2)=-2*t13;           mnc(3,3)=t3/pi;
%
```

```
%
% stiffness matrix
ks(1,1)=wh*wh;
ks(2,2)=ra*ra*wa*wa;
ks(3,3)=rb*rb*wb*wb;
ks=ks*mas
%
%
knc(2,3)=-t15;
knc(3,3)=-t18/pi;
%
%
bs(3,3)=2*rb*rb*wb*damb;
bs=bs*mas
%
%
bnc(1,1)=0;    bnc(1,2)=-pi;         bnc(1,3)=t4;
bnc(2,1)=0;    bnc(2,2)=pi*(a-.5);   bnc(2,3)=-t16;
bnc(3,1)=0;    bnc(3,2)=-t17;        bnc(3,3)=-t19/pi;
%
%
%
r1=[    -2.*pi
     2*pi*(a+0.5)
        -t12];
%
s1=[0  1    t10/pi];
s2=[1 (.5-a)  t11/(2*pi)];
%
%
% Roger approximation of aero forces
rka=[0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.6 0.7 0.8 0.9 1.0
1.2 1.4 1.6 1.8 2.0]';
[m,n]=size(rka);
%
gam1=0.2;  gam2=0.4; gam3=0.6; gam4=0.8;


for ii=1:nn
for jj=1:nn
  tempa=0; tempb=0;
   for jjrk=1:m
   rk=rka(jjrk);
   [fl,gl]=aero(rk,b,mnc,bnc,knc,r1,s1,s2);
   flij=fl(ii,jj);   glij=gl(ii,jj);
al=[ 1                        0
     0                        rk
   -rk*rk                     0
   rk*rk/(rk*rk+gam1*gam1)    rk*gam1/(rk*rk+gam1*gam1)
   rk*rk/(rk*rk+gam2*gam2)    rk*gam2/(rk*rk+gam2*gam2)
```

```
   rk*rk/(rk*rk+gam3*gam3)  rk*gam3/(rk*rk+gam3*gam3)
   rk*rk/(rk*rk+gam4*gam4)  rk*gam4/(rk*rk+gam4*gam4) ]';

     tempa=tempa+al'*al;
     tempb=tempb+al'*[flij  glij]';
     end
xe=inv(tempa)*tempb;
p0(ii,jj)=xe(1);
p1(ii,jj)=xe(2);
p2(ii,jj)=xe(3);
p3(ii,jj)=xe(4);
p4(ii,jj)=xe(5);
p5(ii,jj)=xe(6);
p6(ii,jj)=xe(7);

end
end

% For ploting approximated Aero. Matrix

for jj=1:m
rk=rka(jj);
irk=i*rk;
rkg(jj)=rka(jj);
[fl,gl]=aero(rk,b,mnc,bnc,knc,r1,s1,s2);
zap=p0+p1*irk+p2*irk*irk+p3*irk/(irk+gam1)+p4*irk/(irk+gam2)+p5*irk/(ir
k+gam3)+p6*irk/(irk+gam4);
      y1f(jj)=fl(1,2);
      y1g(jj)=gl(1,2);
y2f(jj)=real(zap(1,2));
y2g(jj)=imag(zap(1,2));
end
%subplot(221),xlabel('Reduced Frequency')
%subplot(221),ylabel('Real of A(1,2)')
%subplot(221),plot(rkg,y1f,'.r',rkg,y2f,'.g')
%subplot(222),xlabel('Reduced Frequency')
%subplot(222),ylabel('Imag. of A(1,2)')
%subplot(222),plot(rkg,y1g,'.r',rkg,y2g,'.g')
%subplot(223),xlabel('Real of A')
%subplot(223),ylabel('Imag of A')
%subplot(223),plot(y1f,y1g,'.r',y2f,y2g,'.g')
%
%pause
%clg

iter=0;
ivf=0;

for kk=1:40;
state=0;
```

```
vel=10.*(kk-1)+10.0;
q=.5*rho*vel*vel;
invms=inv(ms-.5*rho*b*b*p2);

ans1=-invms*(ks-q*p0);
ans2=-invms*(bs-.5*rho*b*vel*p1);
ans3=q*invms*p3;
ans4=q*invms*p4;
ans5=q*invms*p5;
ans6=q*invms*p6;
for ii=1:nn
for jj=1:nn
state(ii,jj+nn)=iden(ii,jj);
state(ii+2*nn,jj+nn)=iden(ii,jj);
state(ii+3*nn,jj+nn)=iden(ii,jj);
state(ii+4*nn,jj+nn)=iden(ii,jj);
state(ii+5*nn,jj+nn)=iden(ii,jj);
state(ii+nn,jj)=ans1(ii,jj);
state(ii+nn,jj+nn)=ans2(ii,jj);
state(ii+nn,jj+2*nn)=ans3(ii,jj);
state(ii+nn,jj+3*nn)=ans4(ii,jj);
state(ii+nn,jj+4*nn)=ans5(ii,jj);
state(ii+nn,jj+5*nn)=ans6(ii,jj);
state(ii+2*nn,jj+2*nn)=-(vel/b)*gam1*iden(ii,jj);
state(ii+3*nn,jj+3*nn)=-(vel/b)*gam2*iden(ii,jj);
state(ii+4*nn,jj+4*nn)=-(vel/b)*gam3*iden(ii,jj);
state(ii+5*nn,jj+5*nn)=-(vel/b)*gam4*iden(ii,jj);
end
end

eig(state);
for ii=1:6*nn;
   if imag(ans(ii)) >= 0.0,
   iter=iter+1;
   vsol(iter)=vel;
   xsol(iter)=real(ans(ii));
   ysol(iter)=imag(ans(ii));


      if(xsol(iter) > 0.0),
      if(ivf==0),
      ivf=1;
      velf=vel;
      end
      end

   end
end
end
vsol=vsol';
```

```
xsol=xsol';
ysol=ysol';
m=size(xsol);
for i=1:m;
result(i,1)=vsol(i);
result(i,2)=xsol(i);
result(i,3)=ysol(i);
end
save open3.out result /ascii ;
figure(1);
plot(xsol,ysol,'or'),xlabel('Real Part'),ylabel('Imag. Part'),grid;
figure(2);
subplot(221),plot(xsol,ysol,'.r'),xlabel('Real Part'),
ylabel('Imag. Part'),grid;
subplot(222);axis('off');
text(0.0,0.85,'Open Loop Flutter Analysis','sc'),
text(0.0,0.75,'3 Degree of Freedom','sc'),
text(0.0,0.65,'Roger Approx.','sc');
subplot(223),plot(vsol,xsol,'.r'),xlabel('Velocity'),
ylabel('Real Part'),grid;
subplot(224),plot(vsol,ysol,'.r'),xlabel('Velocity'),
ylabel('Imag. Part'),grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file : aero.m
% approximation of aero forces
function [f,g]=aero(rk,b,mnc,bnc,knc,r1,s1,s2)
i=sqrt(-1);
cs=theod(rk);
z=(mnc*(i*rk)*(i*rk)+(bnc+cs*r1*s2)*(i*rk)+knc+cs*r1*s1);
z=2*z;
f=real(z);g=imag(z);
%
```

3.5. Aeroservoelastic Design using LQR/LQG

Using rational function approximation described in the previous section, the aeroelastic equation of motion is successfully transformed into the linear time invariant state space equation. The state equation can be expressed into the following form;

$$\begin{Bmatrix} \dot{x}_s \\ \ddot{x}_s \\ \dot{x}_{a3} \\ .. \\ \dot{x}_{aN} \end{Bmatrix} = \begin{bmatrix} 0 & I & 0 & .. & 0 \\ -\overline{M}^{-1}\overline{K} & -\overline{M}^{-1}\overline{B} & q\overline{M}^{-1}P_3 & .. & q\overline{M}^{-1}P_N \\ 0 & I & -\left(\frac{V}{b}\right)\gamma_1 I & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & I & 0 & 0 & -\left(\frac{V}{b}\right)\gamma_{N-2} I \end{bmatrix} \begin{Bmatrix} x_s \\ \dot{x}_s \\ x_{a3} \\ .. \\ x_{aN} \end{Bmatrix} + \begin{bmatrix} 0 \\ \overline{M}^{-1}B_{ACT} \\ 0 \\ 0 \\ 0 \end{bmatrix} \beta \tag{85}$$

or

$$\dot{x} = Ax + Bu \tag{86}$$

where

$$\{B_{ACT}\} = \begin{Bmatrix} 0 \\ 0 \\ \omega_\theta^2 \bar{r}_\theta^2 \end{Bmatrix}$$

Let's design flutter suppression system using the following airfoil data.

$\omega_h = 50.0\ rad/\text{sec},\ \ \omega_\theta = 100.0\ rad/\text{sec},\ \ \omega_\beta = 300.0 rad/\text{sec}$

$a = \text{-}0.4,\ c = 0.6, b = 1, \bar{x}_\theta = 0.2,\ \bar{x}_\beta = 0.0125, \bar{r}_\theta^2 = 0.25,\ \bar{r}_\beta^2 = 0.00625, \mu = 40$

The flutter speed of this model is about 300 ft/sec. Control system design in the state space requires to fix the design airspeed. Usually the design airspeed is fixed as arbitrarily. Let's fix the design airspeed as 320 ft/sec, which is higher than the open loop flutter speed.
At the design airspeed, the system eigenvalues are

```
-14.3870 ± i 339.6737
  5.0715 ± i 70.9743
-25.5913 ± i 74.9236
```

There are several control algorithms are available. Here, let's use the classical LQR (Linear Quadratic Regulator) theory to design a control system.

Consider the linear time invariant state space equation

$$\dot{x} = Ax + Bu$$
$$y = Cx \tag{87}$$

where A is the system matrix, B is the control matrix, and C is the output matrix. LQR theory determines the optimal gain matrix K such that the state-feedback law $u=-Kx$ subject to minimize the quadratic cost function

$$J = \frac{1}{2}\int_0^\infty (x^T Qx + u^T Ru)dt \tag{88}$$

where Q and R are the weighting matrices. The corresponding optimal control is given by

$$u = -Kx = -R^{-1}B^T Px \tag{89}$$

where K is optimal feedback gain matrix, which can be obtained from the Riccati matrix, P. The Riccati matrix is determined by the solution of the following steady state Riccati equation.

$$PA - A^T P + C^T QC - PBR^{-1}B^T P = 0 \tag{90}$$

The closed loop state equation can be written as

$$\dot{x} = (A - BK)x \tag{91}$$

The controller gives the flowing closed loop eigenvalues.

```
-107.2088 ± i 314.2080
  -5.1270 ± i 73.0405
 -49.1710 ± i 64.1661
```

It can be seen that the controller stabilizes the unstable pitching mode. It is assumed that Q and R are identity matrices, and C is defined to feedback the plunge and pitching modes only. Figure shows the closed loop system eigenvalues for different airspeed.

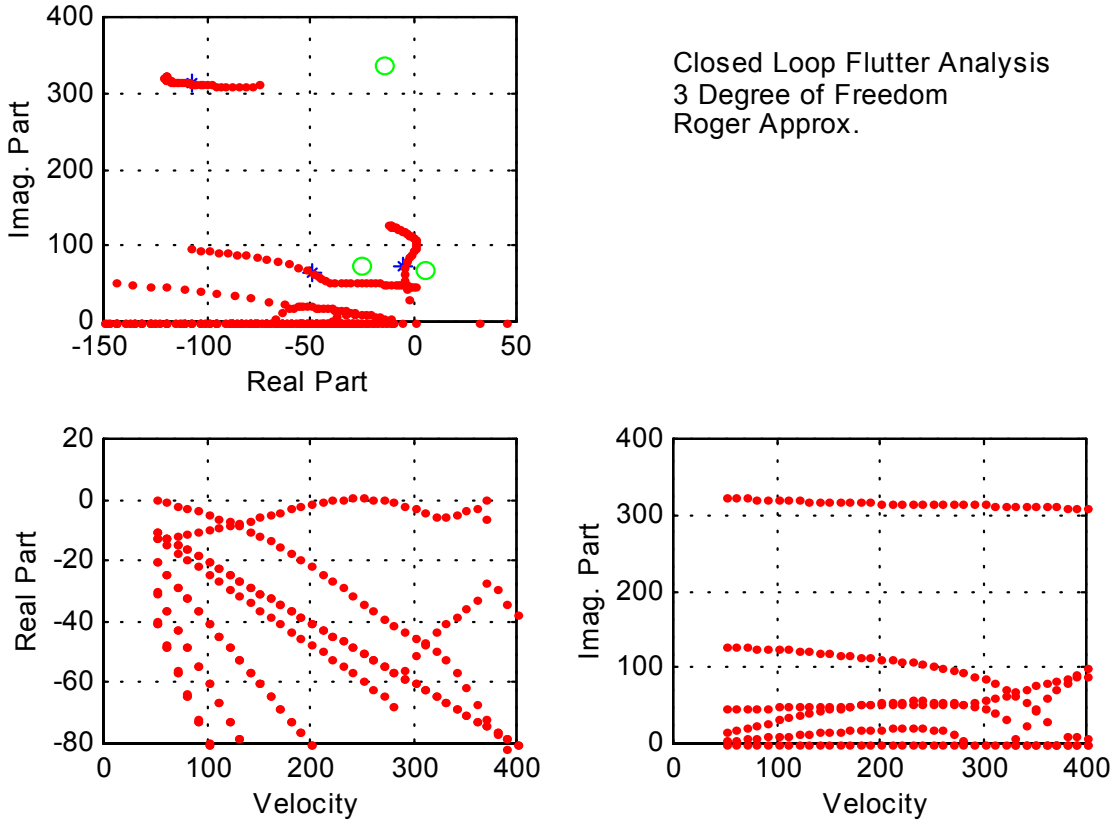


Figure Closed Loop Flutter Analysis Results.

```
close all
clear
% function [result2]= close3(wh,wa,wb,a,c,xa,xb)
% version 1992 1 6
% By nam, changho
% 3 dof system ;roger approx.
wh=50.0;    wa=100.0;     wb=300.0;
a=-0.4;     c=0.5;        b=1;
xa=0.2;     xb=0.0125;
ra=sqrt(0.25);            rb=sqrt(0.00625);
mu=40;

nn=3;i=sqrt(-1);rho=0.002378;
mas=pi*rho*mu*b*b;  damb=0.0;
sqr=sqrt(1-c*c);
arc=acos(c);
%
%
```

```
%
% final version of t function generating the aero matrix
t1=-(2+c*c)/3*sqr+c*arc;
t3=-(1-c*c)/8*(5.*c*c+4)+.25*c*(7+2*c*c)*sqr*arc-(1./8.+c*c)*arc*arc;
t4=c*sqr-arc;
t5=-(1-c*c)-arc*arc+2*c*sqr*arc;
t7=c*(7+2*c*c)/8*sqr-(1/8+c*c)*arc;
t8=-1/3*(1+2*c*c)*sqr+c*arc;
t9=.5*(sqr*(1-c*c)/3+a*t4);
t10=sqr+arc;
t11=(2-c)*sqr+(1-2*c)*arc;
t12=(2+c)*sqr-(1+2*c)*arc;
t13=-.5*(t7+(c-a)*t1);
t15=t4+t10;
t16=t1-t8-(c-a)*t4+0.5*t11;
t17=-2*t9-t1+(a-.5)*t4;
t18=t5-t4*t10;
t19=-.5*t4*t11;
%
%
%
iden=zeros(nn,nn);ks=zeros(nn,nn);knc=zeros(nn,nn);bs=zeros(nn,nn);
for ii=1:nn
   iden(ii,ii)=1.0;
end
% mass matrix
ms(1,1)=1;
ms(1,2)=xa;
ms(1,3)=xb;
ms(2,1)=xa;
ms(2,2)=ra*ra;
ms(2,3)=rb*rb+xb*(c-a);
ms(3,1)=xb;
ms(3,2)=rb*rb+xb*(c-a);
ms(3,3)=rb*rb;
ms=ms*mas
%
%
mnc(1,1)=-pi;
mnc(1,2)=pi*a;
mnc(1,3)=t1;
mnc(2,1)=pi*a;
mnc(2,2)=-pi*(1./8.+a*a);
mnc(2,3)=-2*t13;
mnc(3,1)=t1;
mnc(3,2)=-2*t13;
mnc(3,3)=t3/pi;
%
%
% stiffness matrix
```

```
ks(1,1)=wh*wh;
ks(2,2)=ra*ra*wa*wa;
ks(3,3)=rb*rb*wb*wb;
ks=ks*mas
%
%
knc(2,3)=-t15;
knc(3,3)=-t18/pi;
%
%
bs(3,3)=2*rb*rb*wb*damb;
bs=bs*mas
%
%
bnc(1,1)=0;
bnc(1,2)=-pi;
bnc(1,3)=t4;
bnc(2,1)=0;
bnc(2,2)=pi*(a-.5);
bnc(2,3)=-t16;
bnc(3,1)=0;
bnc(3,2)=-t17;
bnc(3,3)=-t19/pi;
%
%
g=[     0.0
        0.0
        rb*rb*wb*wb];
g=g*mas;
%
%
r1=[    -2.*pi
     2*pi*(a+0.5)
        -t12];
%
%
s1=[ 0
     1
    t10/pi]' ;
%
%
s2=[  1
    (.5-a)
    t11/(2*pi) ]' ;
%
%
%
% Roger approximation of aero forces
rka=[0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.6 0.7 0.8 0.9 1.0
1.2 1.4 1.6 1.8 2.0]';
```

```
[m,n]=size(rka);
%
gam1=0.2;
gam2=0.4;
gam3=0.6;
gam4=0.8;


for ii=1:nn
for jj=1:nn
  tempa=0;
  tempb=0;
   for jjrk=1:m
   rk=rka(jjrk);
   [fl,gl]=aero(rk,b,mnc,bnc,knc,r1,s1,s2);
flij=fl(ii,jj);
glij=gl(ii,jj);
   al=[ 1  0
    0   rk
   -rk*rk   0
   rk*rk/(rk*rk+gam1*gam1)  rk*gam1/(rk*rk+gam1*gam1)
   rk*rk/(rk*rk+gam2*gam2)  rk*gam2/(rk*rk+gam2*gam2)
   rk*rk/(rk*rk+gam3*gam3)  rk*gam3/(rk*rk+gam3*gam3)
   rk*rk/(rk*rk+gam4*gam4)  rk*gam4/(rk*rk+gam4*gam4) ]';


     tempa=tempa+al'*al;
     tempb=tempb+al'*[flij  glij]';
     end
xe=inv(tempa)*tempb;

p0(ii,jj)=xe(1);
p1(ii,jj)=xe(2);
p2(ii,jj)=xe(3);
p3(ii,jj)=xe(4);
p4(ii,jj)=xe(5);
p5(ii,jj)=xe(6);
p6(ii,jj)=xe(7);

end
end

% For ploting approximated Aero. Matrix

for jj=1:m
rk=rka(jj);
irk=i*rk;
rkg(jj)=rka(jj);
[fl,gl]=aero(rk,b,mnc,bnc,knc,r1,s1,s2);
```

```
zap=p0+p1*irk+p2*irk*irk+p3*irk/(irk+gam1)+p4*irk/(irk+gam2)+p5*irk/(ir
k+gam3)+p6*irk/(irk+gam4);
r1f(jj)      =fl(1,1); r1g(jj)=      gl(1,1);
a1f(jj)=real(zap(1,1));a1g(jj)=imag(zap(1,1));
r2f(jj)      =fl(1,2); r2g(jj)=      gl(1,2);
a2f(jj)=real(zap(1,2));a2g(jj)=imag(zap(1,2));
r3f(jj)      =fl(1,3); r3g(jj)=      gl(1,3);
a3f(jj)=real(zap(1,3));a3g(jj)=imag(zap(1,3));
r4f(jj)      =fl(2,1); r4g(jj)=      gl(2,1);
a4f(jj)=real(zap(2,1));a4g(jj)=imag(zap(2,1));
r5f(jj)      =fl(2,2); r5g(jj)=      gl(2,2);
a5f(jj)=real(zap(2,2));a5g(jj)=imag(zap(2,2));
r6f(jj)      =fl(2,3); r6g(jj)=      gl(2,3);
a6f(jj)=real(zap(2,3));a6g(jj)=imag(zap(2,3));
r7f(jj)      =fl(3,1); r7g(jj)=      gl(3,1);
a7f(jj)=real(zap(3,1));a7g(jj)=imag(zap(3,1));
r8f(jj)      =fl(3,2); r8g(jj)=      gl(3,2);
a8f(jj)=real(zap(3,2));a8g(jj)=imag(zap(3,2));
r9f(jj)      =fl(3,3); r9g(jj)=      gl(3,3);
a9f(jj)=real(zap(3,3));a9g(jj)=imag(zap(3,3));
end
figure(11)
plot(r1f,r1g,'*r',a1f,a1g,'og'),xlabel('Real of A(1,1)'),
ylabel('Imag of A(1,1)'),grid on
figure(12)
plot(r2f,r2g,'*r',a2f,a2g,'og'),xlabel('Real of A(1,2)'),
ylabel('Imag of A(1,2)'),grid on
figure(13)
plot(r3f,r3g,'*r',a3f,a3g,'og'),xlabel('Real of A(1,3)'),
ylabel('Imag of A(1,3)'),grid on
figure(14)
plot(r4f,r4g,'*r',a4f,a4g,'og'),xlabel('Real of A(2,1)'),
ylabel('Imag of A(2,1)'),grid on
figure(15)
plot(r5f,r5g,'*r',a5f,a5g,'og'),xlabel('Real of A(2,2)'),
ylabel('Imag of A(2,2)'),grid on
figure(16)
plot(r6f,r6g,'*r',a6f,a6g,'og'),xlabel('Real of A(2,3)'),
ylabel('Imag of A(2,3)'),grid on
figure(17)
plot(r7f,r7g,'*r',a7f,a7g,'og'),xlabel('Real of A(3,1)'),
ylabel('Imag of A(3,1)'),grid on
figure(18)
plot(r8f,r8g,'*r',a8f,a8g,'og'),xlabel('Real of A(3,2)'),
ylabel('Imag of A(3,2)'),grid on
figure(19)
plot(r9f,r9g,'*r',a9f,a9g,'og'),xlabel('Real of A(3,3)'),
ylabel('Imag of A(3,3)'),grid on
```

```
state=0.0;
temp=inv(ms)*g;
% Let's set design airspeed
vel=320.
q=.5*rho*vel*vel;
invms=inv(ms-.5*rho*b*b*p2);
ans1=-invms*(ks-q*p0);
ans2=-invms*(bs-.5*rho*b*vel*p1);
ans3=q*invms*p3;
ans4=q*invms*p4;
ans5=q*invms*p5;
ans6=q*invms*p6;

for ii=1:nn
for jj=1:nn
state(ii,jj+nn)=iden(ii,jj);
state(ii+2*nn,jj+nn)=iden(ii,jj);
state(ii+3*nn,jj+nn)=iden(ii,jj);
state(ii+4*nn,jj+nn)=iden(ii,jj);
state(ii+5*nn,jj+nn)=iden(ii,jj);
state(ii+nn,jj)=ans1(ii,jj);
state(ii+nn,jj+nn)=ans2(ii,jj);
state(ii+nn,jj+2*nn)=ans3(ii,jj);
state(ii+nn,jj+3*nn)=ans4(ii,jj);
state(ii+nn,jj+4*nn)=ans5(ii,jj);
state(ii+nn,jj+5*nn)=ans6(ii,jj);
state(ii+2*nn,jj+2*nn)=-(vel/b)*gam1*iden(ii,jj);
state(ii+3*nn,jj+3*nn)=-(vel/b)*gam2*iden(ii,jj);
state(ii+4*nn,jj+4*nn)=-(vel/b)*gam3*iden(ii,jj);
state(ii+5*nn,jj+5*nn)=-(vel/b)*gam4*iden(ii,jj);
end
g1(ii)=0.0;
g1(ii+nn)=temp(ii);
g1(ii+2*nn)=0.0;
g1(ii+3*nn)=0.0;
g1(ii+4*nn)=0.0;
g1(ii+5*nn)=0.0;
end
g1=g1';
state=state;
%
%
%
iter=0;
eig(state);
  for ii=1:6*nn;
      if imag(ans(ii)) >= 1.0e-4,
         if abs(real(ans(ii))) >= 1.0e-4,
          if real(ans(ii)) <= 50.0,
          if real(ans(ii)) >= -500.0,
```

```
           iter=iter+1;
           xopen(iter)=real(ans(ii));
           yopen(iter)=imag(ans(ii));
          end
          end
      end
      end
  end
xopen=xopen';  yopen=yopen';
%
%
% dx/dt = [state] x + [g1] u
%   y   = C x
%

for i=1:4
for j=1:6*nn
c(i,j)=0.0;
end
end
c(1,1)=1.0;
c(2,2)=1.0;
c(3,4)=1.0;
c(4,5)=1.0;
%
qq=c'*c/10000.0;     rr=0.1;
%qq=qq/10000;rr=rr/10000;
[gain,ricca]=lqr2(state,g1,qq,rr);
%  u = -Kx  minimizes the cost function:
temp= -g1*gain;
iter=0;
eig(state+temp);
  for ii=1:6*nn;
      if imag(ans(ii)) >= 1.0e-4,
      if abs(real(ans(ii))) >= 1.0e-4,
      if real(ans(ii)) <= 50.0,
      if real(ans(ii)) >= -500.0,
      iter=iter+1;
      xclse(iter)=real(ans(ii));
      yclse(iter)=imag(ans(ii));
      end
      end
      end
      end
  end
xclse=xclse';
yclse=yclse';
%
break
%
```

```
iter=0;

for kk=1:41;
vel=10.*(kk-1)+50;
q=.5*rho*vel*vel;
state=0;
invms=inv(ms-.5*rho*b*b*p2);

ans1=-invms*(ks-q*p0);
ans2=-invms*(bs-.5*rho*b*vel*p1);
ans3=q*invms*p3;
ans4=q*invms*p4;
ans5=q*invms*p5;
ans6=q*invms*p6;
for ii=1:nn
for jj=1:nn
state(ii,jj+nn)=iden(ii,jj);
state(ii+2*nn,jj+nn)=iden(ii,jj);
state(ii+3*nn,jj+nn)=iden(ii,jj);
state(ii+4*nn,jj+nn)=iden(ii,jj);
state(ii+5*nn,jj+nn)=iden(ii,jj);
state(ii+nn,jj)=ans1(ii,jj);
state(ii+nn,jj+nn)=ans2(ii,jj);
state(ii+nn,jj+2*nn)=ans3(ii,jj);
state(ii+nn,jj+3*nn)=ans4(ii,jj);
state(ii+nn,jj+4*nn)=ans5(ii,jj);
state(ii+nn,jj+5*nn)=ans6(ii,jj);
state(ii+2*nn,jj+2*nn)=-(vel/b)*gam1*iden(ii,jj);
state(ii+3*nn,jj+3*nn)=-(vel/b)*gam2*iden(ii,jj);
state(ii+4*nn,jj+4*nn)=-(vel/b)*gam3*iden(ii,jj);
state(ii+5*nn,jj+5*nn)=-(vel/b)*gam4*iden(ii,jj);
end
end

state=state+temp;



eig(state);
  for ii=1:6*nn;
      if imag(ans(ii)) >= -1.0e-4,
      if abs(real(ans(ii))) >= 1.0e-4,
      if real(ans(ii)) <= 50.0,
      if real(ans(ii)) >= -500.0,
      iter=iter+1;
      vsol(iter)=vel;
      xsol(iter)=real(ans(ii));
      ysol(iter)=imag(ans(ii));
      end
      end
```

```
      end
      end
  end
%
end
%
          vsol=vsol';
          xsol=xsol';
          ysol=ysol';
          m=size(xsol);
          for i=1:m;
          result2(i,1)=vsol(i);
          result2(i,2)=xsol(i);
          result2(i,3)=ysol(i);
          end
          save close3.out result2 /ascii;
load open3.out
[mopn,nopn]=size(open3);
for i=1:mopn;
vopn(i)=open3(i,1);
xopn(i)=open3(i,2);
yopn(i)=open3(i,3);
end

figure(3);
plot(xsol,ysol,'.r',xopn,yopn,'.g'),
xlabel('Real Part'),ylabel('Imag. Part'),
title('3 Degree of Freedom, Open/Closed Loop System'),grid

%clg;
figure(4);
plot(xsol,ysol,'+r',xopen,yopen,'og',xclse,yclse,'*b'),
axis([-300 50 0 400])
xlabel('Real Part'),ylabel('Imag. Part'),
title('3 Degree of Freedom, Closed Loop System'),grid
axis;
%pause
figure(5);
subplot(221),plot(xopen,yopen,'og',xclse,yclse,'*b',xsol,ysol,'.r'),
axis([-150 50 0 400]),xlabel('Real Part'),ylabel('Imag. Part'),grid;

subplot(222),axis('off');
text(0.0,0.85,'Closed Loop Flutter Analysis','sc'),
text(0.0,0.75,'3 Degree of Freedom','sc'),
text(0.0,0.65,'Roger Approx.','sc');

subplot(223),plot(vsol,xsol,'.r'),
axis([0 400 -80 20]),xlabel('Velocity'),ylabel('Real Part'),grid;

subplot(224),plot(vsol,ysol,'.r'),
```

```
axis([0 400 0 400]),xlabel('Velocity'),ylabel('Imag. Part'),grid;
%pause
```

3.6. A New Flutter Analysis Technique Using Unsteady Aerodynamic Eigen Formulation

The aeroelastic analysis of the wing relies on an accurate representation of the unsteady aerodynamic forces. Since the analysis is faced with the problem of coupling the unsteady aerodynamic representation into the structural response model, this representation also must be in a convenient computational form. The unsteady aerodynamic model is divided into two groups which are frequency domain and time domain analyses. The aeroelastic stability analysis is usually carried out in the frequency domain, such as *V-g* method and *p-k* method. However, in modern aircraft design, the interaction among the structure, aerodynamics and control system can not be neglected. The control system exhibits sufficient gain and phase margins. A difficulty in using modern control theory for the design of systems to control aeroelastic behavior is the requirement of transforming the unsteady aerodynamic forces, normally provided in the frequency domain, into the time domain. The usual procedure for the time invariant state-space form is to approximate the unsteady aerodynamic force represented in the frequency domain as a ration function in the Laplace domain. However, this requires quite amount of CPU time and results in increase in the state due to the augmented aerodynamic states.

In this section, a formulation for the unsteady aerodynamics in time domain is discussed. In structural dynamic analysis, the dynamic behavior of the complex structures is often reduced to a few degrees of freedom using normal mode analysis technique. Recently, researchers expected that these techniques could be applied to unsteady aerodynamic model. With the reduced order models, it might be possible to predict the unsteady aerodynamic response of the system over a wide reduced frequency range. In this section, we will develop an aeroelastic model using an aerodynamic eigen formulation. A reduced-order model will be obtained by expanding the unsteady vortex into a sum of first few aerodynamic eigenmodes. To account for the effects of truncated higher modes, a static correction is included in the model. To get the system matrix with the real

constant, a modal decomposition technique which transforms the aerodynamic matrix into a canonical modal form is introduced.

3.6.a Basic Theory – Time Domain Unsteady Aerodynamics

We divide the bound vortex representing the airfoil into M elements, divide the free vortex representing the traveling wake into (N-M) elements. The total number of vortex elements is N. For an isolated flat airfoil in two-dimensional incompressible flow, the kernel function is

$$K_{ij} = \frac{1}{2\pi(x_{i3/4} - \xi_j)} \tag{92.a}$$

where $x_{i3/4}$ is the location of the i-th collocation, i.e., the three-quarter chord of each element, and ξ_j is the location of the j-th point vortex, i.e., the quarter chord of each element.

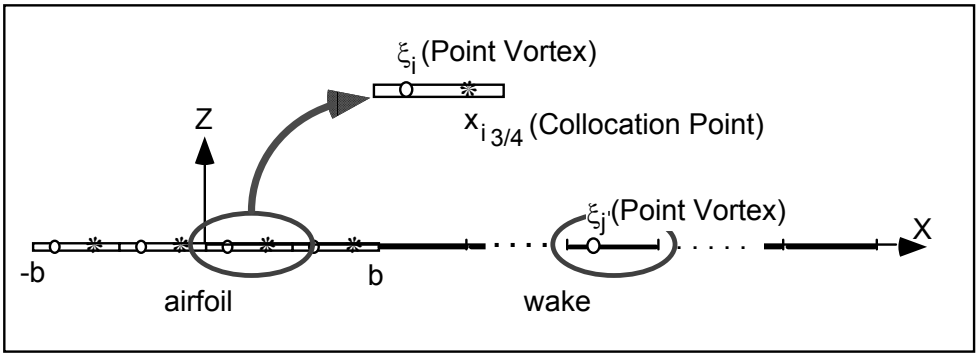


Figure Discrete Vortex Model for 2D Airfoil

For kernel function representation, we express the downwash in the discrete-space domain,

$$\{W_{3/4}\} = \begin{bmatrix}[K_1] & [K_2]\end{bmatrix}\begin{Bmatrix}\{\Gamma_1\}\\ \{\Gamma_2\}\end{Bmatrix} \tag{92.b}$$

where $W_{3/4}$ is a (M×1) vector containing downwashes at three-quarter points of vortex elements of the airfoil. Γ_1, Γ_2 denote the vortex strength on the airfoil, wake, respectively. And K_1, K_2 are the corresponding kernel functions, which are sized M×M and M×(N-M)

For conservation of vorticity,

$$\frac{d}{dt}\int_{-b}^{b}\gamma(x,t)dx = -V\gamma_{TE}$$

where b is the semichord, V is the velocity and γ is the vorticity distribution on the airfoil. In the discrete-space domain,

$$-[S_1]\{\dot{\Gamma}_1\} = \frac{\Gamma_{M+1}}{\Delta x}V \tag{93}$$

with $[S_1] = [1 \quad 1 \quad \ldots \quad 1]$ (1×M).

For the special case of $\Delta x = V\Delta t$, we can write

$$\Gamma_{M+1}^{n+1} + \Gamma_{M+1}^{n} = -2\sum_{i=1}^{M}(\Gamma_i^{n+1} - \Gamma_i^{n})$$

For convection of free wakes,

$$\dot{\Gamma}_i = -V\frac{\partial\Gamma_i}{\partial x} \cong -V\frac{\Gamma_{i-1} - \Gamma_{i+1}}{2\Delta x}, (i = M+2, \ldots, N-2)$$

$$\dot{\Gamma}_{N-1} \cong -V \frac{\Gamma_{N-2} - \Gamma_{N-1}}{\Delta x} \tag{94}$$

$$\dot{\Gamma}_{N} \cong -V \frac{\Gamma_{N-1} + (w-1)\Gamma_{N}}{\Delta x}$$

However, the free wake convection is more conveniently described in discrete time domain as follows.

$$\Gamma_i^{n+1} \cong \Gamma_{i-1}^{n}, (i = M+2, ..., N-1)$$

$$\Gamma_N^{n+1} \cong w\Gamma_N^{n} + w\Gamma_{N-1}^{n}, (0.95 < w < 1) \tag{95}$$

The weighting factor w in Eqs. (94) and (95) is introduced to prevent sudden change in the induced downwash due to the finite length of the wake vortex sheet. By assigning a value near unity, one can accumulate all of the vortex strengths that would exist between the last cutting point and infinity, thereby simulating the effect of these truncated vorticities on the induced downwash. It can be shown that any w value equal to or larger than unity would overestimate this effect. For example, $w=1$ will render all of the truncated vortex strengths to be accumulated in the last vortex sheet element, $i=N$. Since the location of this last vortex ξ_N is closer to a collocation point on the airfoil than any other vortex locations behind, the resulting kernel function will yield an induced downwash with a magnitude greater than that of true one. Hall suggests $0.95 < w < 1$.

Putting Eq(92) (93) and (94) together in matrix form,

$$\begin{bmatrix} 0 & 0 \\ \begin{bmatrix} -S_1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & I_{N-M-1} \end{bmatrix} \end{bmatrix} \begin{Bmatrix} \dot{\Gamma}_1 \\ \dot{\Gamma}_2 \end{Bmatrix} = \begin{bmatrix} [K_1] & [K_2] \\ 0 & \frac{V}{\Delta x}[B_a] \end{bmatrix} \begin{Bmatrix} \Gamma_1 \\ \Gamma_2 \end{Bmatrix} - \begin{Bmatrix} W_{3/4} \\ 0 \end{Bmatrix} \tag{96}$$

where $[I_{N-M-1}]$ is the (N-M-1) × (N-M-1) identity matrix, and $\frac{1}{\Delta x}[B_a]$ matrix approximates the partial derivatives of free vortex elements in x, given in Eq. (94).

Solving for Γ_1 from the top of Eq.(96) gives

$$\{\Gamma_1\} = -[K_1^{-1}K_2]\{\Gamma_2\} + [K_1]^{-1}\{W_{3/4}\}$$

Hence,

$$\left[\begin{bmatrix} -S_1 \\ 0 \end{bmatrix}[K_1^{-1}K_2] + \begin{bmatrix} 0 & 0 \\ 0 & I_{N-M-1} \end{bmatrix}\right]\{\dot{\Gamma}_2\} = \frac{V}{\Delta x}[B_a]\{\Gamma_2\} + \begin{bmatrix} S_1 \\ 0 \end{bmatrix}[K_1]^{-1}\{\dot{W}_{3/4}\} \qquad (97)$$

On the other hand, a discrete version of Eq.(97) can be written as

$$\begin{aligned} &\left[\begin{bmatrix} -2S_1 \\ 0 \end{bmatrix}[K_1^{-1}K_2] + [I_{N-M}]\right]\{\Gamma_2^{n+1}\} = \\ &\left[\begin{bmatrix} -2S_1 \\ 0 \end{bmatrix}[K_1^{-1}K_2] + [C]\right]\{\Gamma_2^{n}\} + \begin{bmatrix} -2S_1 \\ 0 \end{bmatrix}[K_1]^{-1}\{W_{3/4}^{n+1}\} + \begin{bmatrix} 2S_1 \\ 0 \end{bmatrix}[K_1]^{-1}\{W_{3/4}^{n}\} \end{aligned} \qquad (98)$$

where

$$C = \begin{bmatrix} -1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ .. & .. & .. & \dots & .. \\ 0 & 0 & \dots & 1 & w \end{bmatrix}$$

3.6.b Determination of B_a Matrix

B_a can be obtained either from the original continuous model given by Eqs. (93) and (94), or from the discrete model given by Eq. (95). Since the discrete model given by Eq. (95) describes the behavior of trailing wake adequately, it is recommended that the discrete model be converted to a continuous one in the following manner. First, write the continuous model given by Eq. (97) in the form

$$[A_a]\{\dot{\Gamma}_2\} = \frac{V}{\Delta x}[B_a]\{\Gamma_2\} + [C_a]\{\dot{W}_{3/4}\} \qquad (99)$$

and the discrete one given by Eq.(98) as

$$[A_{ad}]\{\Gamma_2^{n+1}\} = [B_{ad}]\{\Gamma_2^{n}\} + [C_{ad}]\{\dot{W}_{3/4}^{n+1}\} \tag{100}$$

where the time step in this discrete model is $\Delta t = \dfrac{V}{\Delta x}$.

It should be emphasized that the downwash velocity vector $\{\dot{W}_{3/4}\}$ acts as an input to the vortex systems. Inverting the homogeneous part of the continuous equation yields,

$$\{\dot{\Gamma}_2\} = \frac{V}{\Delta x}[A_a]^{-1}[B_a]\{\Gamma_2\} \tag{101}$$

By integrating the Eq. (101) from t^n to t^{n+1}, one can relate the discrete model to the continuous model as

$$\{\Gamma_2^{n+1}\} = [A_{ad}]^{-1}[B_{ad}]\{\Gamma_2^{n}\} = \exp\left\{(\frac{1}{\Delta t}[A_a]^{-1}[B_a])\Delta t\right\}\{\Gamma_2^{n}\} \tag{102}$$

from which one obtains

$$[B_a] = [A_a]\ln([A_{ad}]^{-1}[B_{ad}]) \tag{103}$$

Once B_a is determined, the unsteady lift and moment can be obtained as follows:

$$L = \rho V\sum_{j=1}^{M}\Gamma_j + \rho\sum_{j=1}^{M}\sum_{k=1}^{j}\dot{\Gamma}_k\Delta x_j \tag{104}$$

$$M = -\rho V\sum_{j=1}^{M}(\xi_j - ab)\Gamma_j - \rho\sum_{j=1}^{M}(\xi_j - ab)\sum_{k=1}^{j}\dot{\Gamma}_k\Delta x_j \tag{105}$$

where ρ, *ab* denote the air density, distance from the midchord to the elastic axis, respectively.

3.6.c Eigen Analysis of Aerodynamic Model

The homogeneous part of Eq. (99) (or Eq. (101)) allows an eigen analysis of the unsteady vortex equation. Assuming a harmonic solution of the vortex in the from $\Gamma_2 = xe^{\lambda t}$ yields

$$\lambda[A_a]\{x\} = \frac{V}{\Delta x}[B_a]\{x\} \tag{106}$$

The solutions of Eq. (106) for λ are always stable, i.e., the real parts of the eigenvalues are negative. This confirms the fact that the creation of wake vortices through unsteady motion of the airfoil should be dissipative; the wake vortices are formed as a result of bound vortices being shed at the trailing edge through viscous activity in the boundary layer around the airfoil. It is seen that the eigenvalues λ are inversely proportional to the factor *Δx/V*, the time for a free vortex element to travel the distance Δx. Thus, if $\bar{\lambda}_i$ and x_i are an eigenvalue and the corresponding right eigenvector of the system

$$\bar{\lambda}_i[A_a]\{x_i\} = [B_a]\{x_i\} \tag{107}$$

then the corresponding eigenvalue and the eigenvector of the system (106) are $\frac{V}{\Delta x}\bar{\lambda}_i$ and x_i, respectively. More generally, we can write the right eigenvector problem as

$$[A_a][X][\bar{\Lambda}] = [B_a][X]$$

where $\bar{\Lambda}$ is a diagonal matrix containing the eigenvalue of the generalized problem given by (107), and *X* is a matrix whose columns contain the corresponding right eigenvectors. Similarly, for the left eigenvector problem,

$$[A_a]^T[Y][\bar{\Lambda}] = [B_a]^T[Y]$$

The eigenvectors satisfy the orthogonality conditions

$$[Y]^T[A_a][X] = [I]$$

$$[Y]^T[B_a][X] = [\overline{\Lambda}]$$

It can be shown that the eigenvalues of the continuous-time system $\bar{\lambda}_i$ are related to the eigenvalues of the discrete-time system, $\tilde{\lambda}_i$ as

$$\bar{\lambda}_i = \ln(\tilde{\lambda}_i) \tag{108}$$

while the eigenvectors of two systems remain the same. Thus, for our purpose, the calculation of $[B_a]$ matrix given by Eq. (103) and the eigenvalues, eigenvectors of the continuous-time system are all facilitated by obtaining the eigenvalues and eigenvectors of the discrete-time system.

3.6.d System Equations for Aeroservoelastic Analysis

The equations of motion for classical 2-DOF wing model are used to conduct aeroservoelastic analysis.

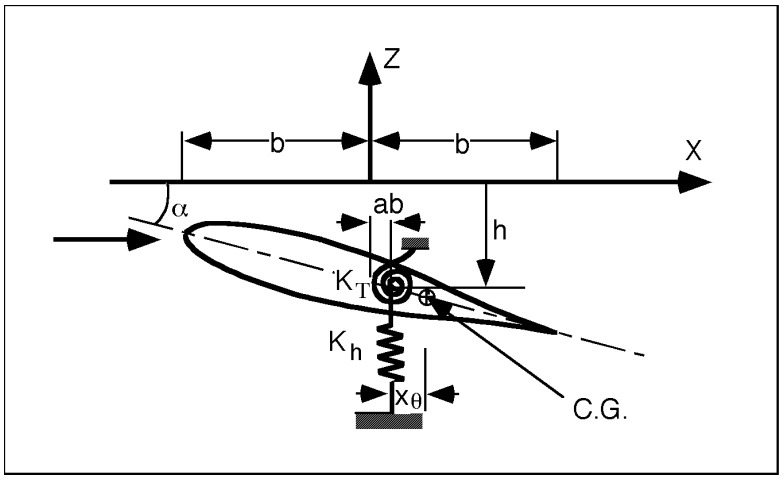


Figure Typical Section Model

For a typical section, the equation of motion is

$$\begin{bmatrix} m & mx_\theta \\ mx_\theta & mr_\theta^{\,2} \end{bmatrix}\begin{Bmatrix} d^2h/dt^2 \\ d^2\alpha/dt^2 \end{Bmatrix}+\begin{bmatrix} K_h & 0 \\ 0 & K_\alpha \end{bmatrix}\begin{Bmatrix} h \\ \alpha \end{Bmatrix}=\begin{Bmatrix} -L \\ M \end{Bmatrix} \tag{109}$$

$$\begin{bmatrix} 1/\bar{r}_\theta^2 & \bar{x}_\theta/\bar{r}_\theta^2 \\ \bar{x}_\theta/\bar{r}_\theta^2 & 1 \end{bmatrix}\begin{Bmatrix} \ddot{\bar{h}} \\ \ddot{\alpha} \end{Bmatrix}+\begin{bmatrix} R^2/\bar{r}_\theta^2 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} \bar{h} \\ \alpha \end{Bmatrix}=\begin{Bmatrix} -Lb/K_\alpha \\ M/K_\alpha \end{Bmatrix}$$

$$[\bar{M}]\begin{Bmatrix} \ddot{\bar{h}} \\ \ddot{\alpha} \end{Bmatrix}+[\bar{K}]\begin{Bmatrix} \bar{h} \\ \alpha \end{Bmatrix}=[\bar{V}]\begin{Bmatrix} \bar{\Gamma}_1 \\ \bar{\Gamma}_2 \end{Bmatrix} \tag{110}$$

where $\omega_h^2 = K_h/m$, $\omega_\theta^2 = K_\alpha/mr_\theta^2$, $\bar{x}_\theta = x_\theta/b$, $\bar{r}_\theta = r_\theta/b$, $\bar{h} = h/b$, $R = \dfrac{\omega_h}{\omega_\theta}$. (•) now denotes a derivative with respect to the nondimensional time $\tau = \omega_\theta t$

And

$$\bar{V} = \frac{1}{\pi\rho\bar{r}_\theta^{\,2}}\begin{bmatrix} -V_\theta F_1 & -F_2 \\ V_\theta G_1 & G_2 \end{bmatrix} = \begin{bmatrix} V_\theta\bar{V}_1 & \bar{V}_2 \end{bmatrix}$$

$$[F_1] = \begin{bmatrix} 1 & 1 & .... & 1 \end{bmatrix}$$
$$[F_2] = \Delta\bar{x}\begin{bmatrix} M & M-1 & ... & 1 \end{bmatrix}$$

$$[G_1] = -\begin{bmatrix} (\bar{\xi}_1 - a) & (\bar{\xi}_2 - a) & .... & (\bar{\xi}_M - a) \end{bmatrix}$$
$$[G_2] = -\Delta\bar{x}\begin{bmatrix} h_1 & h_2 & ... & h_M \end{bmatrix}$$

$$h_k = \sum_{j=k}^{M}\xi_j - (M-k+1)ab$$

where the mass ratio is defined as μ, V_θ is the reduced velocity and $\overline{\Gamma}$ is $\Gamma / b^2 \omega_\theta$

For an airfoil undergoing plunging and pitching motion, the nondimensional downwash vector is related to the plunging and pitching degrees of freedom via

$$\{\overline{W}_{3/4}\} = [E_1]\begin{Bmatrix} \dot{\overline{h}} \\ \dot{\alpha} \end{Bmatrix} + V_\theta [E_2]\begin{Bmatrix} \overline{h} \\ \alpha \end{Bmatrix} \tag{111}$$

$$E_1 = \begin{bmatrix} 1 & (\overline{x}_{3/4} - \overline{e}) \end{bmatrix}, E_2 = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{112}$$

Here $\overline{x}_{3/4}$ and a a are the vectors containing nondimensional locations of the three-quarter points of vortex elements and elastic axis of the airfoil aft of midchord. Combining Eqs. (92), (94), and (110), and eliminating $\overline{\Gamma}_1$ from the equation, we can get the following contracted equations:

$$\begin{bmatrix} [A_a] & V_\theta [A_{c12}] & [A_{c13}] \\ [0] & [I_2] & [0] \\ [A_{c31}] & V_\theta [A_{c32}] & [A_{c33}] \end{bmatrix} \begin{Bmatrix} \dot{\overline{\Gamma}}_2 \\ \ddot{\overline{h}} \\ \dot{\alpha} \\ \ddot{\overline{h}} \\ \ddot{\alpha} \end{Bmatrix} = \begin{bmatrix} \frac{V_\theta}{\Delta \overline{x}} [B_a] & [0] & [0] \\ [0] & [0] & [I_2] \\ V_\theta [B_{c31}] & -[K] + V_\theta^2 [B_{c32}] & V_\theta [B_{c33}] \end{bmatrix} \begin{Bmatrix} \overline{\Gamma}_2 \\ \dot{\overline{h}} \\ \alpha \\ \dot{\overline{h}} \\ \dot{\alpha} \end{Bmatrix} \tag{113}$$

where

$$[A_{c12}] = \begin{bmatrix} -S_1 \\ 0 \end{bmatrix} [\overline{K}_1]^{-1} [E_2],$$

$$[A_{c13}] = \begin{bmatrix} -S_1 \\ 0 \end{bmatrix} [\overline{K}_1]^{-1} [E_1]$$

$$[A_{c31}] = [\overline{V}_2][\overline{K}_1]^{-1}[\overline{K}_2]$$

$$[A_{c32}] = -[\overline{V}_2][\overline{K}_1]^{-1}[E_2]$$

$$[A_{c33}] = [\overline{M}] - [\overline{V}_2][\overline{K}_1]^{-1}[E_1]$$

$$[B_{c31}] = [\overline{V}_1][\overline{K}_1]^{-1}[\overline{K}_2]$$

$$[B_{c32}] = [\overline{V}_1][\overline{K}_1]^{-1}[E_2]$$

$$[B_{c33}] = [\overline{V}_1][\overline{K}_1]^{-1}[E_1]$$

3.6.e Construction of Reduced-Order Model

We decompose the unsteady solution of vortex into two parts; a modal expansion part which is the sum of a few of the aerodynamic eigenmodes, and a static correction part:

$$\{\overline{\Gamma}_2\} \cong [X_R]\{\overline{c}\} + \{\overline{\Gamma}_s\} \tag{114}$$

Here X_R is a (N-M)×N_R matrix where N_R << (N-M) whose columns are the N_R right eigenvectors corresponding to the first N_R eigenvalues of Eq. (106) that are nearest to the origin, $\overline{c}$ is the generalized coordinate vector. The modal expansion part approximates the solution by spanning an N_R-dimensional subspace of the complete (N-M)-dimensional space, and therefore does not include the effects of all the remaining (N-M-N_R) aerodynamic modes. The role of the static correction part $\overline{\Gamma}_s$ is to account for the missing effects by providing at least the quasi-static contribution from the truncated higher modes. It can be shown that the static correction is obtained as

$$\{\overline{\Gamma}_s\} = \{\overline{\Gamma}_{2s}\} - [X_R]\{\overline{c}_s\} \tag{115}$$

Although the reduced-order model can be used to predict the open-loop flutter and divergence by solving the complex generalized eigenvalue problem, it is not appropriate

for a closed-loop control purpose. This is because the some of the entries in the matrix equation are not real but complex due to the existence of the complex eigenvector, X_R.

To overcome such difficulty and make all elements of the matrix equation real, we introduce a modal decomposition technique which transforms the aerodynamic matrix into a canonical modal form where a pair of complex conjugate eigenvalues appear in a (2×2) block on the diagonal and the real eigenvalues appear on the diagonal. For the aerodynamic system Eq. (101) (without the factor of $\frac{V_\theta}{\Delta x}$) that has m pairs of complex conjugate eigenvalues $-\sigma_i \pm j\omega_i$ and n real eigenvalues $-\lambda_i$ (N-M=$2m$+n), the modal matrix is

$$\begin{aligned} [A_{can}] &= [Q]^T([A_a]^{-1}[B_a])[P] \\ [Q]^T[P] &= [I_{N-M}] \end{aligned} \tag{116}$$

For the reduction of order, we decompose the unsteady vortex into

$$\{\bar{\Gamma}_2\} \cong [P_R]\{q\} + \{\bar{\Gamma}_s\} \tag{117}$$

where P_R is a (N-M)×N_R matrix whose columns are the N_R columns of P corresponding to the first N_R eigenvalues that are nearest to the origin, q is the new generalized coordinate vector.

Therefore, the static correction is obtained as

$$\{\bar{\Gamma}_s\} = \{\bar{\Gamma}_{2s}\} - [P_R]\{q\} \tag{118}$$

The first term is the solution of the following system:

$$0 = \frac{V_\alpha}{\Delta \bar{x}}[B_a]\{\bar{\Gamma}_{2s}\} + [C_a]\{\dot{\bar{W}}_{3/4}\} \tag{119}$$

The quasi-static solution of q is

$$\begin{aligned}\{q_s\} &= -\frac{V_\alpha}{\Delta\bar{x}}\left\{[Q_R]^T[A_a^{-1}B_a][P_R]\right\}^{-1}[Q_R]^T[A_a^{-1}C_a]\{\dot{\bar{W}}_{3/4}\} \\ &= -\frac{V_\alpha}{\Delta\bar{x}}[A_{canR}]^{-1}[Q_R]^T[A_a^{-1}C_a]\{\dot{\bar{W}}_{3/4}\}\end{aligned} \tag{120}$$

where A_{canR} is a $N_R \times N_R$ submatrix whose nonzero entries are those of A_{can} corresponding to the N_R retained eigenvalues, and Q_R^T is a matrix whose rows contain the corresponding N_R left eigenvectors.

Therefore,

$$\{\bar{\Gamma}_s\} = \frac{\Delta\bar{x}}{V_\theta}\left\{-[B_a]^{-1} + [P_R][A_{canR}]^{-1}[Q_R]^T[A_a]^{-1}\right\}[C_a]\{\dot{\bar{W}}_{3/4}\} \tag{121}$$

With the static correction as obtained above, a new reduced-order aeroservoelastic model can be obtained:

$$\begin{bmatrix} [I_{NR}] & V_\theta[\bar{A}_{c12}] & [\bar{A}_{c13}] \\ [0] & [I_2] & [0] \\ [\bar{A}_{c31}] & V_\theta[\bar{A}_{c32}] & [\bar{A}_{c33}] \end{bmatrix} \begin{Bmatrix} \dot{q} \\ \dot{\bar{h}} \\ \dot{\alpha} \\ \ddot{\bar{h}} \\ \ddot{\alpha} \end{Bmatrix} = \begin{bmatrix} \frac{V_\theta}{\Delta\bar{x}}[A_{canR}] & [0] & V_\theta[\bar{B}_{c13}] \\ [0] & [0] & [I_2] \\ V_\theta[\bar{B}_{c31}] & -[K]+V_\theta^2[\bar{B}_{c32}] & V_\theta[\bar{B}_{c33}] \end{bmatrix} \begin{Bmatrix} q \\ \bar{h} \\ \alpha \\ \dot{\bar{h}} \\ \dot{\alpha} \end{Bmatrix} + \frac{\Delta\bar{x}}{V_\theta}\begin{bmatrix} [Q_R]^T[K_s][A_{c13}] \\ [0] \\ [A_{c31}][K_s][A_{c13}] \end{bmatrix} \begin{Bmatrix} \dddot{\bar{h}} \\ \ddot{\alpha} \end{Bmatrix} \tag{122}$$

where

$$[\overline{A}_{c12}] = [Q_R]^T [A_a]^{-1} [A_{c12}]$$
$$[\overline{A}_{c13}] = [Q_R]^T ([A_a]^{-1} [A_{c13}] + [A_a]^{-1} [B_a][K_s][A_{c13}] - \Delta\bar{x}[K_s][A_{c12}])$$
$$[\overline{A}_{c31}] = [A_{c31}][P_R], [A_{c32}] = [\overline{A}_{c32}]$$
$$[\overline{A}_{c33}] = [A_{c33}] - \Delta\bar{x}([A_{c31}][K_s][A_{c12}] - [B_{c31}][K_s][A_{c13}])$$
$$[B_{c13}] = -[Q_R]^T [A_a]^{-1} [B_a][K_s][A_{c12}]$$
$$[\overline{B}_{c31}] = [B_{c31}][P_R], [\overline{B}_{c32}] = [B_{c32}]$$
$$[\overline{B}_{c33}] = [B_{c33}] - \Delta\bar{x}([B_{c31}][K_s][A_{c12}])$$
$$[K_s] = -[B_a]^{-1} + [P_R][A_{canR}^{-1}][Q_R]^T [A_a]^{-1}$$

As discussed in this section, static correction is adopted during the order reduction in order to provide the quasistatic contribution from the truncated higher modes. Therefore, the terms containing $(\dddot{\bar{h}} \quad \dddot{\alpha})$ are appeared in the reduced order model. When we design the controller, these terms will not be accounted; they might be assumed as the structural uncertainty. If the magnitude of the structural uncertainty is small, the robust stability of the closed loop system is not degraded even though the controller is designed with the model that neglects the uncertainty. As a matter of fact, it was found that the effects of those terms are negligibly small, these terms are not considered in the control system design. Note that the reduced modes of the aerodynamic model are retained in the reduced order model, and both models (full order model and reduced order model) use same physical coordinates for the structural modes.

It has to be noted that the nondimensional divergence speed of the system, $V_{\theta Div}$, can be obtained by solving the eigenvalue problem of the following equation;

$$\left[-[\overline{K}] \quad V_{\theta Div}^2 [\overline{B}_{c32}]\right] \begin{Bmatrix} \bar{h} \\ \alpha \end{Bmatrix} = \{0\} \tag{123}$$

3.6.f Example - Flutter Analysis Using Eigen Formulation

The typical section model is used in this study as shown in Figure. A model has two degrees of freedom which are plunge and pitch. Linear and torsion springs at the elastic

axis act to restrain motion in plunge and twist. The mass ratio μ is 20, the static imbalance $\bar{x}_\theta$=0.2, the radius of gyration $\bar{r}_\theta$ is 0.5, and the location of the elastic axis aft of midchord a is -0.1. The frequency ratio R of the uncoupled modes is 0.3.

Figure show the discrete time eigenvalues and continuous time eigenvalues for the typical section model, respectively. For this analysis, ten vortex elements are used to model the airfoil. The wake was modelled using 100 vortex elements, and the length of the wake was taken to be 10 chord lengths. As shown in Figure, the magnitude of all the discrete time eigenvalues is less than 1. These eigenvalues are mapped into the left in the continuous time domain, which means that the system is stable.

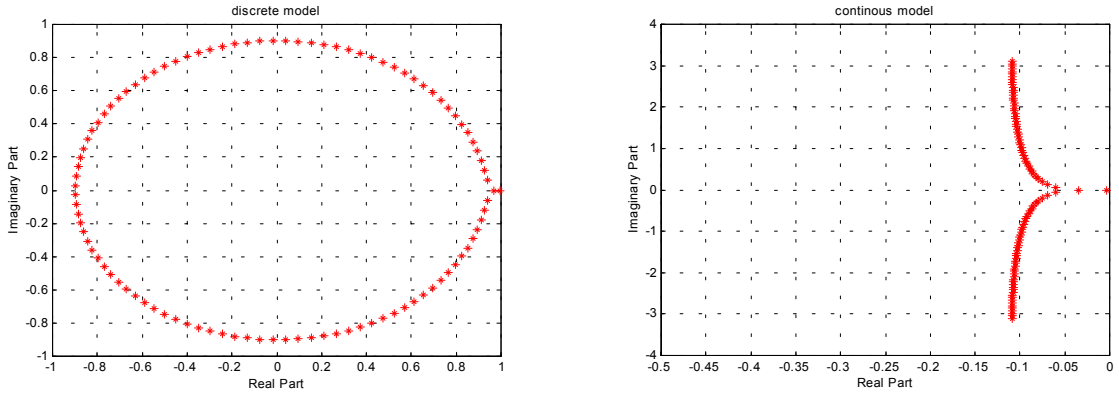


Figure (a) Discrete Time Eigenvalues, (b) Continuous Time Eigenvalues of the Unsteady Aerodynamics

The eigenmode information is used in order to compute the flutter speed in the typical section model. For model reduction, total 20 eigenmodes from 100, which have the nearest continuous time eigenvalues to the origin, are used in the reduced order model along with the static correction.

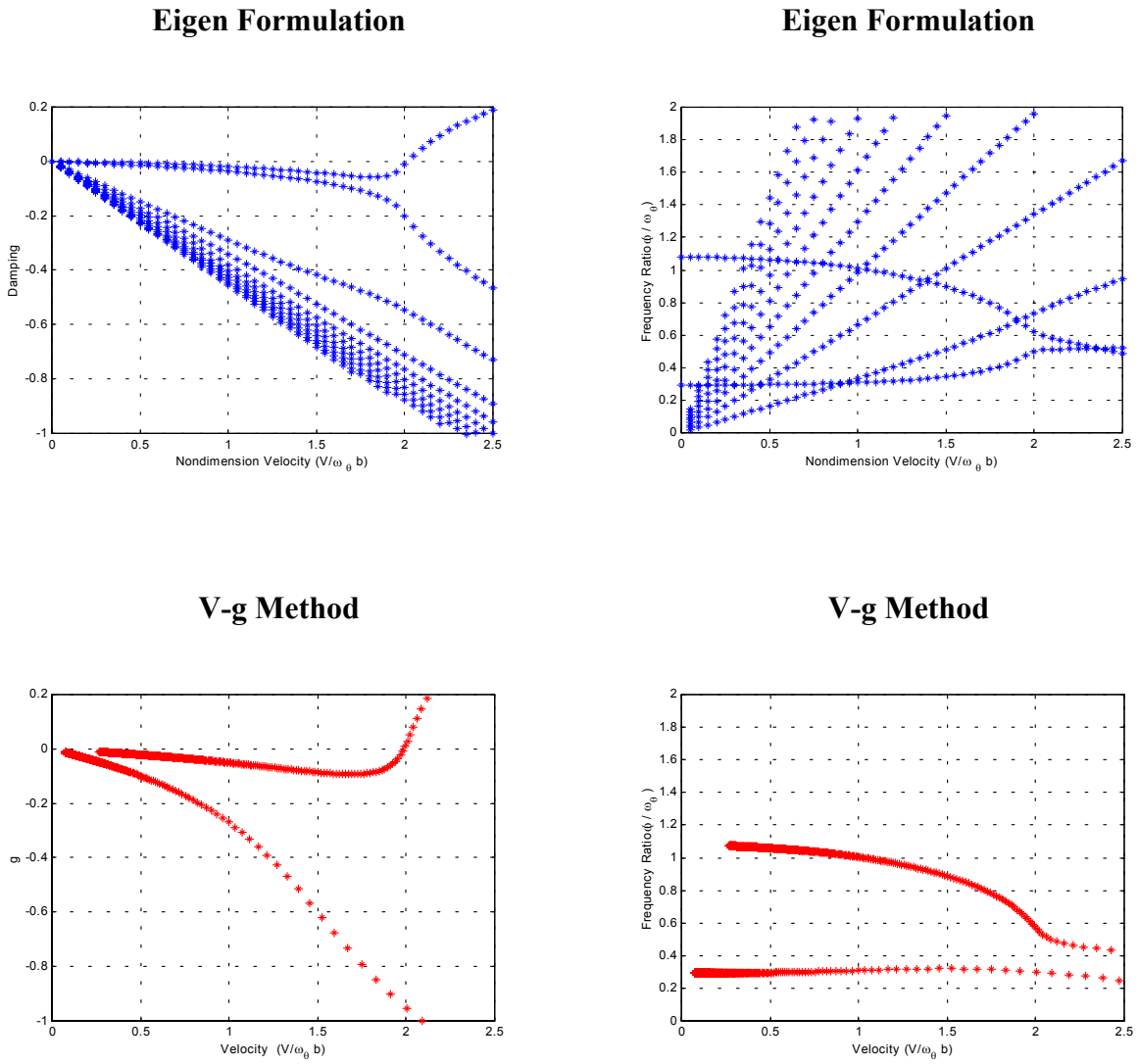


Figure Frequency and Damping Plots of the Typical Section Model with the Variation of the Reduced Velocity

Figure show the frequency(ω/ω_θ), damping of the reduced order model against the reduced velocity($V/\omega_\theta b$). For comparison, the results of V-g method are also plotted.

In *V-g* method, the unsteady aerodynamics formulated by Theodorsen are used. The flutter speed calculated using *V-g* method is about 2.0. Note that the frequency and g from *V-g* method do not have the physical meaning except at the flutter speed. The value of *g* in *V-g* method denotes the damping required for the harmonic motion. On the other hand, the value of *g* obtained from the current method denotes the real damping of the system at the specified airspeed. That is identical with the flutter speed using the current formulation.

It should be noted that his system model is shown in the discrete time domain and has the complex coefficient in the matrix equation. However, in this paper, the equations of motion were derived on the continuous time domain, and the reduced order model with the real constant was proposed using a modal decomposition technique. Therefore, the current formulation can be directly used for the aeroelastic control system design. The predicted divergence speed from V-g method is about 2.5.

Comparing the V-g method with given typical section model, the current method has a disadvantage in the computational time for aerodynamic eigenvalues. However, the situation will be different when the more realistic wing model is used for the aeroelastic analysis. Currently, this method is being extended to analize the aeroservoelastic characteristics of the 3D wing model using the unsteady vortex lattice method. It is expected that there will be substantial CPU time savings compared to the existing method, such as frequency domain analysis with the rational function approximation.

Matlab List

Test.m

```
close all
clear

   e=-0.1;
   xalpha=0.2;ralpha=0.5;
   xmu=20.0;omegah=0.3; alpha=0.996;
%   e=-0.2;
%   xalpha=0.1;ralpha=0.5;
%   xmu=20.0;omegah=0.4; alpha=0.996;
```

```
disp('Aeroservoelastic Design Study Using Eigen Formulation')
 iopen=input('Open loop snalysis ? (y/n)=(1/0)   ');
  if(iopen == 1),
  ui=input('initial vel    '); % nondimensionalized w.r.t (omega_theta
b)
  uf=input('final vel   ');
  delu=input('increment vel   ');
    else
    disp('Fix the design airspeed in order to design a control system')
    u=input('design velocity   ');
    end
n=110;m=10;nm=100;nms=104;
nmode=20;nmd2=24;
numcon=2;numout=4;
xm=m;xn=n;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vortex1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%=====Invert xk1 matrix=====
%
xk1=inv(xk1);
aaa1=zeros(n-m,m);
aaa1(1,1:m)= -s1(1,1:m);
bad=aaa1*xk1*xk2;
%
  aad(1:n-m,1:n-m)=bad(1:n-m,1:n-m);
  bad(1:n-m,1:n-m)=bad(1:n-m,1:n-m)+xc(1:n-m,1:n-m);

  for i=1:n-m
  aad(i,i)=aad(i,i)+1.0d00;
  end
%
  aad=inv(aad);
  ad=aad*bad;
  bd(1,1)=1.0;
%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 d2cmr
 msize=nrdc+4;
%
%     Set desired speed for designing controller
%
  if(iopen == 1)
  opnfltr;
  disp('Open Loop Analysis is Done !')
  break
  end
%
```

Vortex1.m

```
%================================================
%     m: number of vortex elements for airfoil.
%     n: total number of vortex elements for airfoil & free wake.
%     x(i): locations of 3/4 chord points of vortex elements.
%     xi(i): locations of 1/4 chord points of vortex elements.
%     xk: K = {K1 K2] matrix (mxn).
%     xk1: K1 matrix (mxm).
%     xk2: K2 matrix (mx(n-m)).
%     s1: S1 matrix (1xm).
%     xc: C matrix (mx(n-m)).
%================================================
%
dx=2./m;dx1=dx/4.;dx2=3.*dx1;
%
for i=1:n
x(i)=dx*(i-1)+dx2-1.0;xi(i)=dx*(i-1)+dx1-1.0;
end
%
for i=1:m
s1(1,i)=2.0;
 for j=1:n
 xk(i,j)=1./(x(i)-xi(j))/2./pi;
 end
end
xk1=xk(1:m,1:m); xk2(1:m,1:n-m)=xk(1:m,1+m:n);
%
for i=1:n-m
 for j=1:n-m
 xc(i,j)=0.d0;
 end
end
  for i=1:n-m-1
  xc(i+1,i)=1.d0;
  end
xc(1,1)=-1.0;xc(n-m,n-m)=alpha;
```

D2cmr.m

```
npar=nm;  nmpar=nm;
nmode=20;
%
% converting routine by zero-order-hold approximation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 ts=1.0;err1=1.d-7 ;err2=1.d-10 ;
%
% construct system matrix for d2c
%
```

```
 rad=zeros(nmode,nmode);sysmat=zeros(nm,nm);
 sysmat(1:nm,1:nm)=ad(1:nm,1:nm);
 lneig=zeros(nm,nm);
 cw1=zeros(nm,nm);cw2=zeros(nm,nm);
%
[ceig,phi,psi]=eign(sysmat,eye(nm));
figure (1) ;
plot(real(ceig), imag(ceig),'r*'),
xlabel('Real Part'),ylabel('Imaginary Part'),
title('discrete model');grid on
%
%
 for i=1:nm
 lneig(i,i)=log(ceig(i)) ;
 end
%     compute log(sysmat)=phi*log(lneig)*transpose(psi)
%
 cw2=phi*lneig*conj(psi');
%
%
%     error check by observing imaginary parts (should be zero)
%
 errsum=0.d0 ;
 for i=1:nm
 for j=1:nm
 chkimag=imag(cw2(i,j));
 errsum=errsum+chkimag*chkimag/(ts*ts) ;
 end
 end

 errchk=sqrt(errsum);
 if errchk > err1,
 disp('warning: accuracy of d2c conversion may be poor.')
 end

 a=real(cw2)/ts;
 wk1=a;
[ceig2,phi,psi]=eign(wk1,eye(nm));
figure (2) ;
plot(real(ceig2), imag(ceig2),'r*'),
xlabel('Real Part'),ylabel('Imaginary Part'),
title('continous model');grid on;axis([-0.5 0 -4 4])

 wk1=zeros(nm,nm);
 ii=1;
 for kk=1:nm
   wr(ii)= real(ceig2(ii));wi(ii)=imag(ceig2(ii));

   if(abs(wi(ii)) > 0.d0),
     wk1(ii,ii)    = wr(ii);wk1(ii+1,ii+1)= wr(ii);
```

```
     wk1(ii,ii+1)  = wi(ii);wk1(ii+1,ii)  =-wi(ii);
     ii=ii+2;
   else
    wk1(ii,ii)=real(ceig2(ii));
    ii=ii+1;
   end
   if(ii > nm), break; end
 end
rad=zeros(nmode,nmode); rad=wk1(1:nmode,1:nmode);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[vecr,vecl]= rmcomm2(npar,nm,ceig,phi,psi);
 nrdc=nmode;
 rvecr(1:nm,1:nmode)=vecr(1:nm,1:nmode);
 rvecl(1:nm,1:nmode)=vecl(1:nm,1:nmode);
```

Opnfltr.m

```
icheck=0;
jcount=1;
disp('Working ......')
for u=ui:delu:uf
  if(icheck == 0)
   fluttrr;
   ac0(1:msize,1:msize)=ac(1:msize,1:msize);
   bc0(1:msize,1:msize)=bc(1:msize,1:msize);
   icheck=1;
  else
   flutteru;
  end
ac1(1:msize,1:msize)=ac(1:msize,1:msize);
bc1(1:msize,1:msize)=bc(1:msize,1:msize);
alf=eign(bc1,ac1);

 for i=1:msize
  if( (imag(alf(i)) > 0.0) & (imag(alf(i)) < 5.0)),
  xxx=[u, real(alf(i)) imag(alf(i))];
  sol(jcount,1:3)=xxx ;jcount=jcount+1;
  end
 end
end
 figure(3)
 plot(sol(:,1),sol(:,2),'b*'),grid,axis([0 2.5 -1.0 0.2]),
 xlabel('Nondimension Velocity (V/\omega _\theta b)'),
 ylabel('Damping'),
 figure(4)
 plot(sol(:,1),sol(:,3),'b*'),grid,axis([0 2.5 0 2.0])
 xlabel('Nondimension Velocity (V/\omega _\theta b)'),
 ylabel('Frequency Ratio (\omega / \omega _\theta)'),
```

```
figure(5)
plot(sol(:,2),sol(:,3),'b*'),grid,
```

Chapter 4
Composite Box Beam Modeling

In this chapter, aeroelastic analysis will be discussed of the high aspect ratio wing. Wing can be modeled as a box beam with an assumption that the chordwise segments of the wing are rigid. Wing skin is covered with a composite laminated plate. In this chapter, we assume that the readers are familiar with the composite materials and the laminated composite.

4.1 Deformation of the High Aspect Ratio Composite Wing

The wing structure is idealized as a box beam whose flexibility is due entirely to the presence of upper and lower laminated cover sheets. It is also assumed that the chordwise sections are rigid so that the wing deformation is a function of the spanwise coordinate. Since the lamina are constrained to act as a unit, in a manner prescribed by the Euler-Bernoulli deformation assumptions, the equivalent bending and torsion stiffness may be computed.

The displacement of the box beam model can be represented by a bending deflection h (positive, upward) along the spanwise axis and a rotation, α (postive, nose-up) about that axis.

$$w(x, y) = h(y) - x\alpha(y) \tag{1}$$

The reference axis is placed at the midchord of the airfoil section along the span. And it is assumed that the upper and lower cover sheets are oriented symmetrically with respect to the middle surface. In the case of slender swept composite beam of high aspect ration, an effective root can be assumed as shown in figure. The y-axis is placed on the midchord along the spanwise direction.

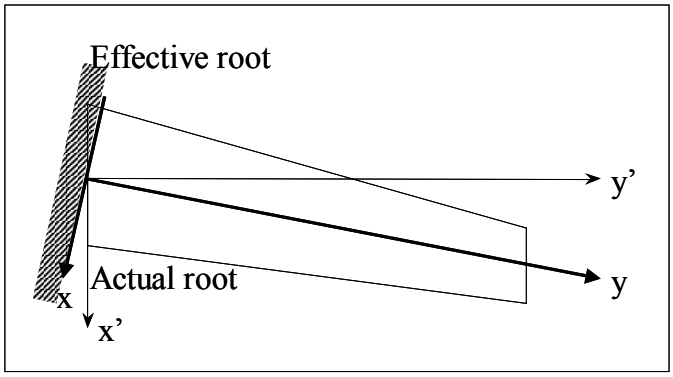


Figure High Aspect Ratio Wing Model

The constitutive relations of the composite lamina can be written as,

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{Bmatrix} \tag{2}$$

where $\overline{Q}_{ij}$ is the stiffness matrix of the lamina. Note that the stiffness matrix is function of the ply orientation of lamina.

The moment resultants are given as the sum of the stress times the area over which they act multiplied by the moment arm with respect to the midplane, $M_{ij} = \int \sigma_{ij} z dz$

Due to the symmetry with respect to the midplane, the moment equation can be reduced as

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \begin{bmatrix} D_{11} & D_{12} & D_{16} \\ D_{12} & D_{22} & D_{26} \\ D_{16} & D_{26} & D_{66} \end{bmatrix} \begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} \tag{3}$$

where D_{ij} is the bending stiffness of the laminated composite. It has to be noted that M_{ij} is the moment per unit width of the box beam along the midplane.

The curvature, κ is expressed in terms of the displacement of box beam as

$$\begin{Bmatrix} \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{Bmatrix} = -\begin{Bmatrix} \dfrac{\partial^2 w}{\partial x^2} \\ \dfrac{\partial^2 w}{\partial y^2} \\ \dfrac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} \tag{4}$$

The bending stiffness D_{ij} is define as

$$D_{ij} = \int \overline{Q}_{ij} z^2 dz = \sum_{p=1}^{N} \overline{Q}_{ij}^{(p)} \frac{1}{3}(t_p^3 + 3t_p z_p^2 + 3z_p t_p^2) \tag{5}$$

where t_p is the thickness, z_p is the coordinate of lower surface for the each lamina. Since we are modeling a box beam, $M_x=0$ and the wing bending moment, M (positve upward), and the torsion moment, T (positive nose-up), will have the following relations.

$$\begin{aligned} M &= -cM_y \\ T &= 2cM_{xy} \end{aligned} \tag{6}$$

where c is the chord of the box beam segment. And the curvature will be expressed in terms of the following variables.

$$\begin{aligned} \kappa_y &= -\frac{\partial^2 w}{\partial y^2} = -h'' \\ \kappa_{xy} &= -2\frac{\partial^2 w}{\partial x \partial y} = 2\alpha' \end{aligned} \tag{7}$$

The equation for the bending and torsion moments of a composite box beam model will be

$$\begin{Bmatrix} -M/c \\ T/2c \end{Bmatrix} = \begin{bmatrix} D_{22} & D_{26} \\ D_{26} & D_{66} \end{bmatrix} \begin{Bmatrix} -h'' \\ 2\alpha' \end{Bmatrix} \tag{8}$$

$$\begin{aligned} \begin{Bmatrix} M \\ T \end{Bmatrix} &= \begin{bmatrix} cD_{22} & -2cD_{26} \\ -2cD_{26} & 4cD_{66} \end{bmatrix} \begin{Bmatrix} h'' \\ \alpha' \end{Bmatrix} \\ &= \begin{bmatrix} EI & -K \\ -K & GJ \end{bmatrix} \begin{Bmatrix} h'' \\ \alpha' \end{Bmatrix} \end{aligned} \tag{9}$$

where *EI* is the bending stiffness, *GJ* is torsional stiffness and *K* is the coupling between beinding and torsion stiffnesses. The *K* parameter provides the elastic coupling between bending and torsion not found in metallic beam analysis. Since *K* depends upon both the stacking sequence of the laminate and the lamina fiber orientation angles, and opportunity to tailor the wing structure to provide advantageous bending-torsion coupling is present.

By taking the inverse of the above equation, the flexibility relations are obtained.

$$\begin{aligned}\begin{Bmatrix} h'' \\ \alpha' \end{Bmatrix} &= \begin{bmatrix} EI & -K \\ -K & GJ \end{bmatrix}^{-1} \begin{Bmatrix} M \\ T \end{Bmatrix} \\ &= \frac{1}{EI \cdot GJ - K^2} \begin{bmatrix} GJ & K \\ K & EI \end{bmatrix} \begin{Bmatrix} M \\ T \end{Bmatrix} \\ &= \begin{bmatrix} \dfrac{1}{EI(1-kg)} & \dfrac{g}{EI(1-kg)} \\ \dfrac{g}{EI(1-kg)} & \dfrac{1}{GJ(1-kg)} \end{bmatrix} \begin{Bmatrix} M \\ T \end{Bmatrix} \end{aligned} \tag{10}$$

where $k=K/EI$, $g=K/GJ$.

4.2 Finite Element Formulation of Box Beam Model

In this chapter, the finite element method will be used to develop an equation of motion for aeroservoelastic analysis. The total strain energy of a laminated beam may be written as follows;

$$U = \frac{1}{2}\int_0^L [EI(h'')^2 + GJ(\alpha')^2 - 2Kh''\alpha']dy \tag{11}$$

The static equilibrium equation of the box beam are derived from the energy equation. The coupled differential equations of the composite box beam model is expressed as

$$\begin{aligned}[EI \cdot h'' - K \cdot \alpha']'' &= 0 \\ [-K \cdot h'' - GJ \cdot \alpha']' &= 0\end{aligned} \tag{12}$$

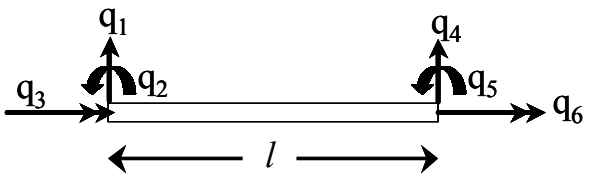


Figure Beam Finite Element Model (6DOF/element)

Now, it is necessary to express displacement, h and rotation, α in terms of the nodal displacements. The bending displacement and rotation of the beam element can be obtained by applying the boundary conditions at each end of the beam element. The boundary conditions are as follows;

At $y=0$,

$$q_1 = h|_{@\, y=0},\ q_2 = h'|_{@\, y=0},\ q_3 = \alpha|_{@\, y=0}$$

At $y=l$,

$$q_4 = h|_{@\, y=l},\ q_5 = h'|_{@\, y=l},\ q_6 = \alpha|_{@\, y=l}$$

By applying the boundary conditions, the bending displacement, h and rotation, α can be expressed in terms of the known nodal displacements, $q_1, \ldots, q_6$.

$$\begin{aligned} h(y) &= \sum_{i=1}^{6} \psi_i(y) \cdot q_i \\ \alpha(y) &= \sum_{i=1}^{6} \varphi_i(y) \cdot q_i \end{aligned} \tag{13}$$

where ψ_i, φ_i are called as shape functions, and defined as

$$\psi_i = \begin{Bmatrix} 1-3\left(\frac{y}{l}\right)^2+2\left(\frac{y}{l}\right)^3 \\ y-\frac{2y^2}{l}+\frac{y^3}{l^2} \\ 0 \\ 3\left(\frac{y}{l}\right)^2-2\left(\frac{y}{l}\right)^3 \\ -\frac{y^2}{l}+\frac{y^3}{l^2} \\ 0 \end{Bmatrix} \tag{14}$$

$$\varphi_i = \begin{Bmatrix} 6g\left(-\frac{y}{l^2}+\frac{y^2}{l^3}\right) \\ 3g\left(-\frac{y}{l}+\left(\frac{y}{l}\right)^2\right) \\ 1-\frac{y}{l} \\ 6g\left(\frac{y}{l^2}-\frac{y^2}{l^3}\right) \\ 3g\left(-\frac{y}{l}+\left(\frac{y}{l}\right)^2\right) \\ \frac{y}{l} \end{Bmatrix} \tag{15}$$

The beam element has six degrees of freedom where q_i are the nodal displacements. With these displacement and rotation functions, the shear force, V bending moment, M and torsional moment, T at each end of the beam element also can be expressed in terms of the nodal displacements.

$$\begin{aligned} V_{y=0} &= EI \cdot h'''_{y=0} - K \cdot \alpha''_{y=0} \\ M_{y=0} &= EI \cdot h''_{y=0} - K \cdot \alpha'_{y=0} \\ T_{y=0} &= GJ \cdot \alpha'_{y=0} - K \cdot h''_{y=0} \\ V_{y=l} &= EI \cdot h'''_{y=l} - K \cdot \alpha''_{y=l} \\ M_{y=l} &= EI \cdot h''_{y=l} - K \cdot \alpha'_{y=l} \\ T_{y=l} &= GJ \cdot \alpha'_{y=l} - K \cdot h''_{y=l} \end{aligned} \tag{16}$$

In matrix form , the static equilibrium equation can be written as

$$[K]\{q\} = \{Q\} \tag{17}$$

where *{Q}* is the element force vector, *[K]* is the element stiffness matrix and *{q}* is the nodal displacement vector.

The components of stiffness matrix are given as;

```
      ke(1,1)= 12.0*EI*(1.0-k*g)/el^3;
      ke(1,2)= 6.0*EI*(1.0-k*g)/el^2;
      ke(1,3)= 0.0;      ke(1,4)=-ke(1,1);
      ke(1,5)= ke(1,2);  ke(1,6)= 0.0;

      ke(2,1)= ke(1,2);
      ke(2,2)= EI*(1.0+3.0*(1.0-k*g))/el;
      ke(2,3)=-K/el;     ke(2,4)=-ke(1,2);
      ke(2,5)=-EI*(1.0-3.0*(1.0-k*g))/el;    ke(2,6)=-ke(2,3);

      ke(3,1)= ke(1,3);  ke(3,2)= ke(2,3);   ke(3,3)= GJ/el;
      ke(3,4)= 0.0;      ke(3,5)=-ke(2,3);   ke(3,6)=-ke(3,3);

      ke(4,1)= ke(1,4);  ke(4,2)= ke(2,4);   ke(4,3)= ke(3,4);
      ke(4,4)= ke(1,1);  ke(4,5)=-ke(1,2);   ke(4,6)= 0.0;

      ke(5,1)= ke(1,5);  ke(5,2)= ke(2,5);   ke(5,3)= ke(3,5);
      ke(5,4)= ke(4,5);  ke(5,5)= ke(2,2);   ke(5,6)= ke(2,3);

      ke(6,1)= ke(1,6);  ke(6,2)= ke(2,6);   ke(6,3)= ke(3,6);
      ke(6,4)= ke(4,6);  ke(6,5)= ke(5,6);   ke(6,6)= ke(3,3);
Note : el is element length
```

The kinetic energy of the composite wing will be written as

$$\begin{aligned} T &= \frac{1}{2}\iiint \rho \dot{w}^2 dxdydz \\ &= \frac{1}{2}\int_0^l \overline{m}(\dot{h} - x_\alpha \dot{\alpha})^2 dy + \frac{1}{2}\int_0^l I_o \dot{\alpha}^2 dy \end{aligned} \tag{18}$$

where $\overline{m}$ is the mass of the wing cross section per span, x_α is the distance from the reference axis to the mass center of the wing cross section, and I_o is the mass moment of inertia of the wing cross section about the center of mass.

The kinetic energy can also be expressed in terms of the nodal displacements.

$$T = \frac{1}{2}\int_0^l \overline{m}\left(\sum_{i=1}^{6}\psi_i(y)\cdot\dot{q}_i - x_\alpha\sum_{i=1}^{6}\varphi_i(y)\cdot\dot{q}_i\right)^2 dy + \frac{1}{2}\int_0^l I_o\left(\sum_{i=1}^{6}\varphi_i(y)\cdot\dot{q}_i\right)^2 dy$$
$$= \frac{1}{2}\lfloor\dot{q}\rfloor\int_0^l \overline{m}\{\psi_i(y) - x_\alpha\varphi_i(y)\}\cdot\lfloor\psi_j(y) - x_\alpha\varphi_j(y)\rfloor dy\{\dot{q}\} \tag{19}$$
$$+ \frac{1}{2}\lfloor\dot{q}\rfloor\int_0^l I_o\{\varphi_i(y)\}\cdot\lfloor\varphi_i(y)\rfloor dy\{\dot{q}\}$$

The moment of inertia about mass center, I_o can be written as

$$I_o = I_\alpha - \overline{m}x_\alpha^2 \tag{20}$$

where $I_\alpha = \overline{m}r_\alpha^2$ is the mass moment of inertia about the reference axis and r_α^2 is the radius of gyration.

The kinetic energy equation can be rewritten as

$$T = \frac{1}{2}\lfloor\dot{q}\rfloor[M]\{\dot{q}\} \tag{21}$$

where [M] is the element mass matrix.

$$\begin{aligned} M_{ij} &= \int_0^l \overline{m}\{\psi_i(y) - x_\alpha\varphi_i(y)\}\cdot\lfloor\psi_j(y) - x_\alpha\varphi_j(y)\rfloor dy \\ &\quad + \int_0^l I_o\{\varphi_i(y)\}\cdot\lfloor\varphi_i(y)\rfloor dy \\ &= \int_0^l \overline{m}\{\psi_i(y) - x_\alpha\varphi_i(y)\}\cdot\lfloor\psi_j(y) - x_\alpha\varphi_j(y)\rfloor dy \\ &\quad + \int_0^l \overline{m}(r_\alpha^2 - x_\alpha^2)\{\varphi_i(y)\}\cdot\lfloor\varphi_i(y)\rfloor dy \end{aligned} \tag{22}$$

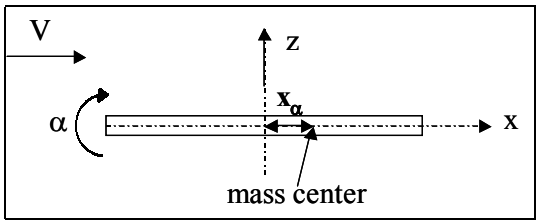


Figure wing cross section

The element of the mass matrix is expressed as

$$M_{ij} = \frac{\overline{m}l}{420}\overline{M}_{ij} \tag{23}$$

The component of the $\overline{M}_{ij}$ is as follows;

```
me(1,1)= 156.0+rg*(420.0*xa/el+504.0*rg*ra^2/el^2);
me(1,2)= 22.0*el+rg*(147.0*xa+252.0*rg*ra^2/el);
me(1,3)=-147.0*xa-210.0*rg*ra^2/el;    me(1,4)=54.0-504.0*rg^2*ra^2/el^2;
me(1,5)=-13.0*el+rg*(63.0*xa+252.0*rg*ra^2/el);
me(1,6)=-63.0*xa-210.0*rg*ra^2/el;

me(2,1)= me(1,2);     me(2,2)=4.0*el^2+rg*(42.0*xa*el+126.0*rg*ra^2);
me(2,3)=-21.0*xa*el-105.0*rg*ra^2;
me(2,4)= 13.0*el+rg*(63.0*xa-252.0*rg*xa^2/el);
me(2,5)=-3.0*el^2+126.0*rg^2*ra^2;   me(2,6)=-14.0*xa*el-105.0*rg*ra^2;

me(3,1)= me(1,3);    me(3,2)=me(2,3);    me(3,3)=140.0*ra^2;
me(3,4)=-63.0*xa+210.0*rg*ra^2/el;    me(3,5)=14.0*xa*el-105.0*rg*ra^2;
me(3,6)= 70.0*ra^2;

me(4,1)= me(1,4);    me(4,2)=me(2,4);     me(4,3)=me(3,4);
me(4,4)= 156.0+rg*(-420.0*xa/el+504.0*rg*ra^2/el^2);
me(4,5)=-22.0*el+rg*(147.0*xa-252.0*rg*ra^2/el);
me(4,6)=-147.0*xa+210.0*rg*ra^2/el;

me(5,1)= me(1,5);     me(5,2)=me(2,5);     me(5,3)=me(3,5);
me(5,4)= me(4,5);
me(5,5)= 4.0*el^2+rg*(-42.0*xa*el+126.0*rg*ra^2);
me(5,6)= 21.0*xa*el-105.0*rg*ra^2;

me(6,1)=me(1,6);    me(6,2)=me(2,6);    me(6,3)=me(3,6);
me(6,4)=me(4,6);    me(6,5)=me(5,6);    me(6,6)=140.0*ra^2;
```

The equation of motion of the composite box beam for the vibration analysis is expressed in the following matrix form:

$$\begin{aligned} [M]\{\ddot{q}\}+[K]\{q\}&=\{0\} \\ -\omega^2[M]\{q\}+[K]\{q\}&=\{0\} \end{aligned} \tag{24}$$

First, the free vibration analysis will be conducted and the equations of motion for flutter analysis will be expressed in terms of the modal coordinates, $\{\xi\}$.

$$[\overline{M}]\{\ddot{\xi}\}+[\overline{K}]\{\xi\}=\{0\} \tag{24}$$

where

$$\begin{aligned} [\overline{M}]&=[\Phi]^T[M][\Phi] \\ [\overline{K}]&=[\Phi]^T[K][\Phi] \end{aligned} \tag{25}$$

$[\Phi]$ is the eigen matrix.

4.3 Unsteady Aerodynamic Forces

In order to calculate the unsteady aerodynamic coefficients for the three dimensional wing model, incompressible 2 dimensional unsteady strip theory can be used. The strip theory defines lift and moment forces along the midchord and acting on sections perpendicular to the mid-chord line. As discussed in Chapter 3, the flow field is represented by a noncirculatory component and a circulatory component. The vertical displacement of the wing segment is expressed as

$$w(x,y,t) = h(y,t) - x\alpha(y,t)$$

where h is the displacement of the midchord (positive up), α is the rotation of a wing section measured perpendicular to the midchord (positive nose up), and x is the distance from the reference axis (aft positive). The lift and moment are written in terms of a circulatory part and non-circulatory part. Using modified strip theory, outlined by Yates, the lift (positive up) per unit length is given as

$$L = -\pi\rho b^2\{\ddot{h} + V_n\dot{\alpha} + V_n\dot{\sigma}\tan\Lambda - ba(\ddot{\alpha} + V_n\dot{\tau}\tan\Lambda)\}_{nc} - \{C_{l\alpha,n}\rho V_n bC(k)Q\}_c \qquad (26)$$

where $\sigma = \dfrac{dh}{dy}$, $\tau = \dfrac{d\alpha}{dy}$

The subscript nc indicates the circulatory part and c indicates the circulatory part of the unsteady aerodynamic forces. And a is location of the reference axis with respect to the midchord, as a fraction of the semi-chord b and ρ is the density, b is the semi chord, V_n (=VcosΛ) is the airspeed measured perpendicular to the midchord, and Λ is the sweep angle. $C_{l\alpha,n}$ is the section lift coefficient and Λ is the sweep angle. And C(k) is the Theoderson function. Q is the downwash distribution defined by

$$Q = \{\dot{h} + V_n\alpha + V_n\sigma\tan\Lambda - b(\frac{C_{l\alpha,n}}{2\pi} + a_{cn} - a)(\dot{\alpha} + V_n\tau\tan\Lambda)\}$$

where a_{cn} is the location of the aerodynamic center with respect to the mid-chord as a fraction of the semichord, b.

{Ref: Yates, E.C., "Calculation of Flutter Characteristics for Finite-Span Swept or Unswept Wings at Subsonic and Supersonic Speeds by a Modified Strip Analysis," NACA RM L57110, March 1958.}

Similiary, the pitching moment (positive nose down) per unit length about the midchord is

$$T = -\pi\rho b^4\{(\frac{1}{8}+a^2)(\ddot{\alpha}+V_n\dot{\tau}\tan\Lambda)+\pi\rho b^2V_n(\dot{h}+V_n\sigma\tan\Lambda)+\pi\rho b^3a(\ddot{h}+V_n\dot{\sigma}\tan\Lambda)$$
$$+\pi\rho b^2{V_n}^2(\alpha-ab\tau\tan\Lambda)\}_{nc}-2\pi\rho b^2V_n[\{\frac{1}{2}-(a-a_{cn})C(k)\frac{C_{l\alpha,n}}{2\pi}\}Q]_c \quad (27)$$

The virtual work due to a distributed lift and moment located at the reference axis is written as

$$\delta W=\int_0^l(\int_0^c p\delta w dx)dy=\int_0^l(L\delta h-T\delta\alpha)dy=\sum_j Q_j\delta q_j \quad (28)$$

where p is the pressure distribution, Q_j is the generalized aerodynamic forces, and q_j is the generalized displacement.

For calculation of subsonic, compressible unsteady aerodynamic forces on harmonically oscillating lifting surfaces, the most generally used scheme is the Doublet Latice Method (DLM). The DLM has proven useful for both planar and nonplanar configurations of multiple interacting lifting surfaces of fairly general geometry. In DLM, the surfaces are represented by a grid of boxes of traperzoidal shape. At the quarter chord of each box is located a line of pressure doublets with a constant strength which is the unsteady lift per unit length of the line. The normalwash at the control points for all boxes in the gird is computed using the kernel function representation of Watkins et. al.{Ref: Watkins, C., Runyan, H., Woolston, D., "On the Kernel Function of the Integral Equation Relating the Lift and Downwash Distributions of Oscillating Finite Wings in Subsonic Flow," NACA Report 1234, 1955.} The control points are at the 3/4 chord point, mid-span for each box.

It is assumed that the flow is irrotational and inviscid so that the flow meets the Kutta

conditions (i.e. finite velocity at the edge) to get around d'Alembert's paradox. Also there is no seperation.

The upwash normal to an oscillating lifting surface is expressed as {ref : Albano, E. and Rodden, W.P., "A Doublet Lattice Method for Calculating Lift Distributions on Oscillating Surfaces in Subsonic Flows," AIAA Journal, Vol 7, Feb. 1969, pp279-285.}

$$\frac{w(x_c, y_c)}{U_\infty} = \frac{1}{8\pi}\iint K(x_c - x_0, y_c - y_0, k, M)\Delta C_p(x_0, y_0)dx_0 dy_0 \qquad (29.a)$$

where k is the reduced frequency (=ωb/U_∞) and M is the Mach number. K is called Kernel function, which is actually the velocity field at (x_c,y_c) due to an elemental normal force at (x_0, y_0).

The above integral equation can be approximated as follows;

$$\frac{w(x_c, y_c)}{U_\infty} = \frac{1}{8\pi}\sum_n (\Delta C_p)_n \iint_{element,n} K(x_c - x_0, y_c - y_0, k, M)dx_0 dy_0 \qquad (29.b)$$

The integration of Kernel function in the streamwise direction is done simply by lumping the effect into a loaded line at the ¼ chord line of the element. The results is an unsteady horseshoe vortex whose bound position lies along the ¼ chord line of the element. There is one control or receiving point per element and the surface normalwash boundary condition is satisfied at each of these points. The control point is centered spanwise on the ¾ chord line of the element. The upwash equation can be written in matrix form;

$$\{\overline{w}\}_{3/4} = [AIC]\{\Delta C_p\} \qquad (30)$$

In the matlab code attached at this book, DLM is used to calculated the unsteady aerodynamic coefficient matrix, [AIC] and the result is saved for the further reduction into the modal coordinate system.

The virtual work due to aerodynamic forces can be rewritten as follow;

$$\begin{aligned}\delta W_A &= q_d \lfloor \delta w \rfloor_{c/4} [S]\{\Delta C_p\} \\ &= q_d \lfloor \delta w \rfloor_{c/4} [S][AIC]^{-1} \{\overline{w}\}_{3c/4}\end{aligned} \tag{31}$$

q_d is the dynamic pressure, a diagonal matrix [S] denotes the panel area. And $\lfloor \delta w \rfloor_{c/4}$, $\{\overline{w}\}_{3c/4}$ denote the virtual displacement at 1/4 chord, the downwash at 3/4 chord of the panel, respectively. With harmonic motion assumption, the downwash is expressed as the following;

$$\{\overline{w}\}_{3c/4} = [\alpha_{ij}]_{3c/4}\{q\} - i\frac{\omega}{V}[W_{ij}]_{3c/4}\{q\} \tag{32}$$

where

$$\begin{aligned}\left[\alpha_{ij}\right]_{3c/4} &= -\left[\frac{\partial Z_j}{\partial x}\right]_{i(3c/4)} \\ \left[W_{ij}\right]_{3c/4} &= \left[Z_j\right]_{i(3c/4)}\end{aligned} \tag{33}$$

Z_j is the displacement function in z-direction and {q} is the generalized coordinate. Similarly, the virtual displacement is written as

$$\{\overline{w}\}_{c/4} = [Z_j]_{i(c/4)}\{q\} \tag{34}$$

Then, the virtual work becomes

$$\begin{aligned}\delta W_A &= q_d \{\delta q\}\lfloor Z_j \rfloor^T_{i(c/4)} [S][AIC]^{-1}\left[[\alpha_{ij}]_{3c/4} - i\frac{\omega}{V}[W_{ij}]_{3c/4}\right]\{q\} \\ &= q_d \{\delta q\}[A]\{q\}\end{aligned} \tag{35}$$

It has to be noted that the aerodynamic forces, [A] is function of the reduced frequency and Mach number.

An executable code, unsdlm.exe is provided for the calculation of the unsteady aerodynamic coefficient. This code will need the input data file, unsdlm.dat.

List of unsdlm.dat : This input data is used in example. For input variables, refer figure.

```
%%%  Title : text
 Composite wing with I/O Control Surfaces(DLM-GEN)
%%%  warea,swep,taper,aspct,y1bar,y2bar,l1bar,l2bar,ftc1,ftc2
```

```
1800.0  25.0  0.8  2.48  0.15  0.45  0.25  0.3  0.25  0.25
%%%  mach,nk,nctotal,nstotal
0.0     20    6    15
%%%  (freq(i),  i=1,nk)
0.8       0.7       0.65      0.6       0.55      0.5
0.45      0.4       0.35      0.3       0.275     0.25
0.225     0.2       0.175     0.15      0.125     0.1
0.05      0.0
% End of DATA
NOTE:
warea : wing area
swep : sweep angle(Deg)
taper : taper ratio
aspct : aspect ratio
y1bar : (inboard location/span) of the 1st control surface
l1bar : (length/span) of the 1st control surface
ftc1  : control surface-to-chord ratio of the 1st control surface
y2bar : (inboard location/span) of the 2nd control surface
l2bar : (length/span) of the 2nd control surface
ftc2  : control surface-to-chord ratio of the 2nd control surface
mach : mach number
nk : no. of redeuced frequencies
nctotal : no. of panels in chordwise direction
nstotal : no. of panels in spanwise direction
freq(1~nk) : reduced frequencies

If model does not have control surfaces set ftc1, ftc2 to 0.2 !!
```

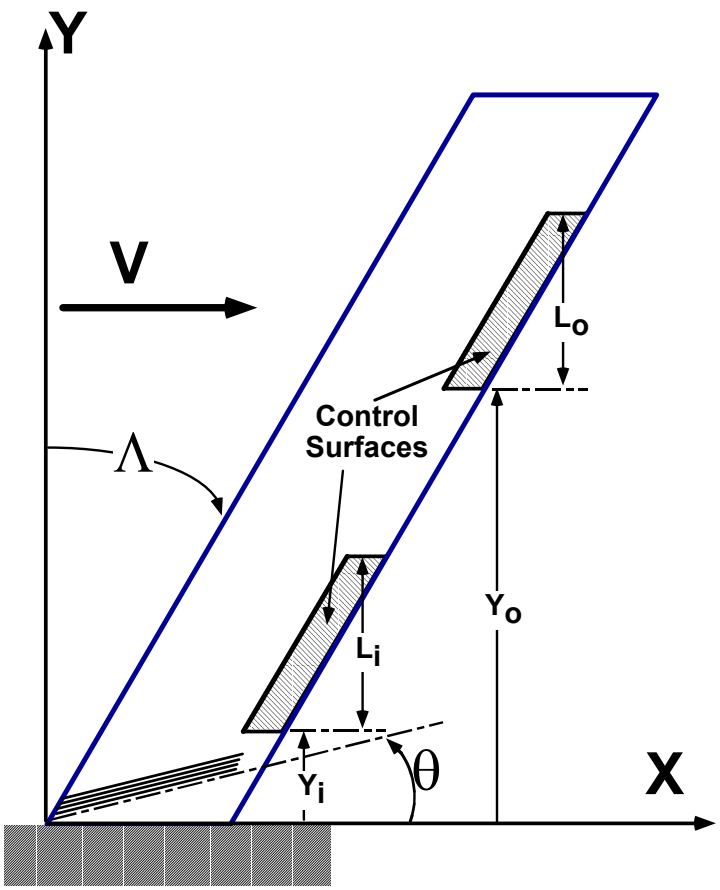


Figure . Composite wing model with control surfaces.

Matlab file, aero.m, will read the aerodynamic coefficient for various reduced frequencies, and calculate the generalized aerodynamic forces in the modal coordinate system using the procedure that we discussed in this section.

List of aero.m

```
%
function [ictrl,jnk,b0,rka,raero] = aero(igust,ndof,need)
%     reads the unsteady aerodynamic influence coefficient matrix
%     supplied by a doublet lattice method
%
%     computes the generalized force matrix in modal coordinates
%     using doublet lattice aerodynamics
%
%
fid_16= fopen('aeroshp.out','w');
file_20 = fopen(['dlmaero.mat']);
deg2rad=360.0d0/(2.0d0*pi);
hhc=zeros(100,10);
jk=sqrt(-1);
```

```
ictrl = fscanf(file_20,'%d',1);

  msize=need;
  if(igust == 0), msize1=msize+ictrl; end
  if(igust == 1), msize1=msize+ictrl+1; end

if(ictrl ~= 0),
for i=1:ictrl,
ipnl(i) = fscanf(file_20,'%d',1);
npnl=ipnl(i);
tmp= fscanf(file_20,'%d',npnl);
ipanel(i,1:npnl)=tmp';
%
%     xhng, yhng : hinge locations of the control surfaces
%
%    x1: x-coord of leading edge/root of the wing.
%    x2: x-coord of trailing edge/root of the wing.
%    x3: x-coord of leading edge/tip of the wing.
%    x4: x-coord of trailing edge/tip of the wing.
%    y1: y-coord of the wing root.
%    y2: y-coord of the wing tip.
tmp= fscanf(file_20,'%g',6);
x1=tmp(1);x2=tmp(2);x3=tmp(3);x4=tmp(4);y1=tmp(5);y2=tmp(6);
tmp= fscanf(file_20,'%g',2);
xhng(i)=tmp(1);yhng(i)=tmp(2);
%
%
  swpl=atan((x3-x1)/(y2-y1));
  swpt=atan((x4-x2)/(y2-y1));
  aswp(i)=(swpl+(swpt-swpl)*xhng(i))*deg2rad;
  xl=x1+(x3-x1)*yhng(i);
  xt=x2+(x4-x2)*yhng(i);
  xhng(i)=xl+(xt-xl)*xhng(i);
  yhng(i)=y1+(y2-y1)*yhng(i);
end
end
%
rkk2 = 1.;
tmp= fscanf(file_20,'%g',2);
nchrd=tmp(1);nspan=tmp(2);
tmp= fscanf(file_20,'%g',3);
ncll=tmp(1);jnk=tmp(2);b0=tmp(3);
b0 = b0/2.;
%
%
tmp= fscanf(file_20,'%g',2*ncll);
for m=1:ncll, xb(m)=tmp(2*m-1);yb(m)=tmp(2*m);end
% upwash & nosedown twist (h,r) w.r.t. 3/4 chord
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [h,r]=tranm(fid_16,msize,ncll,yb,xb);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for i = 1:ncll
 for j = 1:msize
 tpr(i,j)=r(i,j)+jk*h(i,j)/b0 ;
 end
```

```
 hh(i)=xb(i)/b0;
 end
%
%     for each control surface find the upward displacement
%
 if(ictrl ~= 0),
  for ic1=1:ictrl
  npnl=ipnl(ic1);
  cswp=cos(aswp(ic1)/deg2rad);  sswp=sin(aswp(ic1)/deg2rad);
   for ic2=1:npnl
   icp=ipanel(ic1,ic2);
   hhc(icp,ic1)=sswp*(xb(icp)-xhng(ic1))-cswp*(yb(icp)-yhng(ic1));
   end
  end
 end
tmp= fscanf(file_20,'%g',ncll);
for m=1:ncll, xb(m)=tmp(m);end
%
% upwash & nosedown twist (h,work) w.r.t. 1/4 chord
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [h,work]=tranm(fid_16,msize,ncll,yb,xb);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tmp= fscanf(file_20,'%g',ncll);
for m=1:ncll, ar(m)=tmp(m); end
%
for nt=1:jnk
rk2= fscanf(file_20,'%g',1);
rka(nt)=rk2;
tpr=real(tpr)+jk*imag(tpr)*(rk2/rkk2);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf(fid_16,'reduced frequency %g \n',rk2);
 fprintf(fid_16,'the upwash due to elastic mode\n');
 fprintf(fid_16,'%10.5e  %10.5e  %10.5e  %10.5e  %10.5e  %10.5e  %10.5e
%10.5e  %10.5e  %10.5e  \n',tpr);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    control mode
%
 if(ictrl ~= 0),
  for ic1=1:ictrl,
  cswp=cos(aswp(ic1)/deg2rad);  npnl=ipnl(ic1);
   for  ic2=1:npnl,
   icp=ipanel(ic1,ic2);  rr1=-cswp;aa1=rk2/b0*hhc(icp,ic1);
   tpr(icp,msize+ic1)=(rr1+jk*aa1);
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf(fid_16,'reduced frequency %g \n',rk2);
 fprintf(fid_16,'the upwash due to the control surface no %g \n',ic1);
 fprintf(fid_16,'%10.5e  %10.5e  %10.5e  %10.5e  %10.5e  %10.5e  %10.5e
%10.5e  %10.5e  %10.5e\n',tpr(:,msize+ic1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  end
 end
%
```

```
%     gust
%   **********************************
%   tpr is the upwash due to the gust
%   **********************************
 if(igust == 1),
  for  k =1: ncll,
  gcos=cos((rk2)*hh(k));gsin=sin((rk2)*hh(k));
  tpr(k,msize1)=gcos-jk*gsin;
  end
  %%%%%%%%%%%%%%%%%%%%
fprintf(fid_16,'reduced frequency %g \n',rk2);
  fprintf(fid_16,'the upwash due to the gust\n');
  fprintf(fid_16,'%10.5e    %10.5e    %10.5e    %10.5e    %10.5e    %10.5e
%10.5e  %10.5e  %10.5e  %10.5e\n',tpr(:,msize1));
end
%
for  i = 1: ncll,

tmp= fscanf(file_20,'%g',2*ncll);
for m=1:ncll, a(m)=tmp(2*m-1)+jk*tmp(2*m);end

 for j =1:msize1,
 hsb(i,j) = 0;
  for k=1:ncll, hsb(i,j)=hsb(i,j)+a(k)*tpr(k,j);end
 end
  for  j=1:msize1, hsb(i,j)=hsb(i,j)*ar(i);end
end
%
for i=1:msize,
   for j=1:msize1,
   hat(j) =0;
      for k =1:ncll, hat(j)=hat(j)+h(k,i)*hsb(k,j);end
   end
   for j=1:msize1,
   cain(i,j)=-hat(j);
   raero(nt,i,j)=cain(i,j);
 end
end
%
%
rkk2 = rk2;
end
fclose(file_20);
fclose(fid_16);
%
```

4.4 Mathematical Aeroservoelastic Model

The goal of this chapter is to conduct an aeroservoelastic wing planform design by using the control surfaces to suppress flutter and to reduce the gust response of the wing.

The equations of motion for aeroservoelastic analysis can be written as

$$[M]\{\ddot{q}\}+[C]\{\dot{q}\}+[K]\{q\}+[M_c]\{\ddot{\delta}_c\}=[Q_a]\{q\}+[Q_c]\{\delta_c\}+[Q_G]\frac{w_G}{V}$$
$$=q_d\left([A_a]\{q\}+[A_c]\{\delta_c\}+[A_G]\frac{w_G}{V}\right) \tag{36}$$

where $\{\ddot{q}\}$ are the generalized modal coordinates, q_d is the dynamic pressure. The matrices $[Q_a]$, $[Q_c]$ and $[Q_G]$ are the generalized aerodynamic forces due to flexible modes, control surfaces deflection and gust, respectively. After vibration analysis, a modal reduction is performed using the selected elastic modes.

The aerodynamic forces are approximated as the transfer functions of the Laplace variable by a least square curve fit approximation in order to define the aeroservoelastic equations of motion in a linear time invariant state-space form. As discussed in the Chapter 3, Roger's method approximates the unsteady aerodynamic forces in the following form.

$$[A_{ap}]=[A_\xi\ A_c\ A_g]=[\overline{P}_0]+[\overline{P}_1]s'+[\overline{P}_2]s'^2+\sum_{j=3}^{N}\frac{[\overline{P}_j]s'}{s'+\gamma_{j-2}} \tag{37}$$

where $\overline{P}=[P_\xi P_c P_g]$, $s'=ik=i\omega b/V=sb/V$ and s is the Laplace variable, k is the reduced frequency, b is the semi-chord and V is the airspeed. The subscripts ξ, c and g indicate elastic, control surface and gust modes, respectively and γ are the aerodynamic poles that are usually preselected in the range of reduced frequencies of interest. In this study, four terms of the aerodynamic poles are included for the rational functions approximation. The augmented aerodynamic state is defined as follows.

$$\{x_{ja}\}=\frac{s'}{s'+\gamma_{j-2}}[P_{j\xi}P_{jc}P_{jg}]\begin{Bmatrix}\xi\\ \delta_c\\ w_g\end{Bmatrix},\ \text{j=3,4,5,6.} \tag{38}$$

It has to be noted that P_{2g} is set to be zero to avoid $\ddot{w}_g$ term in the state equation. Therefore,

$$[A_g]=[P_{0s}]+[P_{1s}]s'+\sum_{j=3}^{N}\frac{[P_{js}]s'}{s'+\gamma_{j-2}} \tag{39}$$

Using the rational function approximation, the system equations of motion is written in the following form.

$$\{\dot{x}_s\} = [A_s]\{x_s\} + [B_s]\{u_s\} + [B_G]\{w_G\} \tag{40}$$

where

$$[A_s] = \begin{bmatrix} 0 & I & 0 & .. & 0 \\ -\tilde{\tilde{M}}^{-1}\bar{K} & -\tilde{\tilde{M}}^{-1}\bar{B} & q_d\tilde{\tilde{M}}^{-1}I & .. & q_d\tilde{\tilde{M}}^{-1}I \\ 0 & P_{3\xi} & -\left(\frac{V}{b}\right)\gamma_1 I & .. & 0 \\ .. & .. & .. & .. & 0 \\ 0 & P_{N\xi} & 0 & 0 & -\left(\frac{V}{b}\right)\gamma_{N-2}I \end{bmatrix}$$

$$[B_s] = \begin{bmatrix} 0 & 0 & 0 \\ q_d\tilde{\tilde{M}}^{-1}P_{0c} & q_d\tilde{\tilde{M}}^{-1}P_{1c}\left(\frac{b}{V}\right) & q_d\tilde{\tilde{M}}^{-1}P_{2c}\left(\frac{b}{V}\right)^2 \\ 0 & P_{3c} & 0 \\ .. & .. & .. \\ 0 & P_{Nc} & 0 \end{bmatrix}$$

$$[B_G] = \begin{bmatrix} 0 & 0 & 0 \\ q_d\tilde{\tilde{M}}^{-1}P_{0g} & q_d\tilde{\tilde{M}}^{-1}P_{1g}\left(\frac{b}{V}\right) & q_d\tilde{\tilde{M}}^{-1}P_{2g}\left(\frac{b}{V}\right)^2 \\ 0 & P_{3g} & 0 \\ .. & .. & .. \\ 0 & P_{Ng} & 0 \end{bmatrix}$$

$$\tilde{\tilde{M}} = \bar{M} - q_d P_{2\xi}\left(\frac{b}{V}\right)^2$$

$$\tilde{\tilde{B}} = \bar{B} - q_d P_{1\xi}\left(\frac{b}{V}\right)$$

$$\tilde{\tilde{K}} = \bar{K} - q_d P_{0\xi}$$

$$\{x_s\} = \left\lfloor \xi^T \ \dot{\xi}^T \ x_a^T \right\rfloor^T$$

$$\{u_s\} = \lfloor \delta_{c1}\ \dot{\delta}_{c1}\ \ddot{\delta}_{c1}\ \delta_{c2}\ \dot{\delta}_{c2}\ \ddot{\delta}_{c2} \rfloor^T$$

$$\{w_G\} = \lfloor w_g, \dot{w}_g \rfloor^T$$

In the above equations, x_a denotes the aerodynamic states and δ_c is the control surface deflection. The control surfaces actuator transfer functions can be expressed in a state space form as follows.

$$\begin{aligned} \{\dot{x}_c\} &= [A_c]\{x_c\} + \{B_c\}\{\delta_{aileron}\} \\ \{u_s\} &= [C_c]\{x_c\} \end{aligned} \tag{41}$$

The actuator transfer function for each of the control surfaces is fixed as follows.

$$\frac{\{\delta_c\}}{\{\delta_{aileron}\}} = \frac{a}{s+a}\frac{a_0}{s^2 + a_1 s + a_0} \tag{42}$$

The gust state space model is included for random gust response calculations. The vertical gust is modeled by a second order Dryden model;

$$\frac{w_g}{w} = \sigma_{wg}\frac{\sqrt{\frac{3V}{L}}\left(s + \frac{V}{L\sqrt{3}}\right)}{\left[s + \frac{V}{L}\right]^2} \tag{43}$$

where σ_{wg} is the RMS value of the gust velocity, L is the characteristic gust length and V is the airspeed. When the low pass filter (a/(s+a)) is included, the state space equation of the gust is expressed as follows.

$$\begin{aligned} \begin{Bmatrix} \dot{x}_{g1} \\ \dot{x}_{g2} \\ \dot{x}_{g3} \end{Bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ -\tau_g^{-2} & -2\tau_g^{-1} & a \\ 0 & 0 & -a \end{bmatrix} \begin{Bmatrix} x_{g1} \\ x_{g2} \\ x_{g3} \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ \tau_g^{-\frac{3}{2}}\sigma_{wg} \end{Bmatrix} w \\ \{w_G\} = \begin{Bmatrix} \alpha_g \\ \dot{\alpha}_g \end{Bmatrix} &= \begin{bmatrix} 1 & \sqrt{3}\tau_g & 0 \\ -\sqrt{3}\tau_g^{-1} & \left(1 - 2\sqrt{3}\right)\sqrt{3}\tau_g a \end{bmatrix} \begin{Bmatrix} x_{g1} \\ x_{g2} \\ x_{g3} \end{Bmatrix} \end{aligned} \tag{44}$$

where $\tau_g = \frac{L}{V}$.

Or

$$\begin{aligned}\{\dot{x}_g\} &= [A_g]\{x_g\} + \{B_g\}w \\ \{w_G\} &= [C_g]\{x_g\}\end{aligned} \tag{45}$$

By including the gust dynamics system and the actuator system for the control surface, the following state space aeroservoelastic model is obtained.

$$\begin{aligned}\{\dot{x}\} &= [A]\{x\} + [B]\{u\} + \{B_w\}w \\ \{y\} &= [C]\{x\}\end{aligned} \tag{46}$$

where $\{x\}^T = \lfloor x_s^T \, x_c^T \, x_g^T \rfloor$. The resulting state vector is consists of elastic modes, aerodynamic states, actuator states and 3 gust states including low pass fiter.

4.5 Controller Design - Pole placement technique

For a given design airspeed, a controller for active flutter suppression is designed. The performance index value (control effort) is used as a measure of control performance. The following output feedback control law is introduced.

$$\{u\} = -\lfloor K_G \rfloor \{y\} \tag{47}$$

where $\lfloor K_G \rfloor$ denotes output feedback gain and {y} includes only structural information. The eigenvalue problem of the closed-loop system can be written as

$$\left([A] - [B][K_G][C]\right)\{\phi^c\}_i = -\lambda_i^c \{\phi^c\}_i \tag{48}$$

where $\{\phi^c\}_i$ is the eigenvector of the closed-loop system corresponding to the eigenvalue λ_i^c. The eigenvalues of the structural modes can be assigned to desired values by using a pole placement technique.

The problem can be stated as a nonlinear parameter optimization problem in which it is necessary to impose specified eigenspace equality constraints and the elements of output feedback gain matrix are the parameters to be determined. Defining $\{p\}$ as the parameter vector that consists of the elements of output feedback gain $\lfloor K_G \rfloor$, the nonlinear programming problem can be formulated as follows.

Determine output feedback gain parameter {p}
subject to

$$f_i(\{p\}) = \lambda_i^d - \lambda_i(\{p\}) = 0, i = 1,2,.., Ns$$

where $f_i(\{p\})$ is the equality constraint equation that is related to the closed-loop eigenvalue assignment, N_s is the number of structural modes and λ_i^d denotes the desired eigenvalues of these modes.

The above problem can be solved by using any gradient-based nonlinear programming algorithm. The homotopic nonlinear programming with minimum norm correction algorithm is used. To enhance convergence, the linear homotopy map is generated. The original eigenvalue assignment problem is replaced by the one-parameter α family as follows.

$$\begin{aligned} f_i(\{p\}) &= \alpha\lambda_i^d + (1-\alpha)\lambda_i(\{p_{start}\}) - \lambda_i(\{p\}) \\ &= 0, \\ 0 \le \alpha &\le 1 \end{aligned} \tag{49}$$

where $\{p_{start}\}$ is the initial starting value of the parameter vector. Sweeping α using a suitably small increment generates a sequence of neighboring problems. These sequence of problems are solved using the neighboring converged solutions to generate starting iteratives for each subsequent problem. The desired solution is reached once solution for α=1 is obtained. More details about this algorithm can be found in chapter 2.

The control performance canbe investigated by comparing a quadratic type energy function

$$J = \int_0^\infty \left(\{y\}^T [Q]\{y\} + \{u\}^T [R]\{u\} \right) dt \tag{50}$$

where the positive definite weight matrices [Q] and [R] should be related to make the integrands correspond to physical energy measures. Note that the first term in J is related to the error energy of the structural modes and the second term is related to the control energy. This objective function is obtained as follows

$$J = tr([P][X_0]) \tag{51}$$

where $[X_0]$ is an initial autocorrelation of the state and [P] is the solution of the following Lyapunov equation.

$$[P][A_c] + [A_c]^T [P] + [C]^T \left([Q] + [K_G]^T [R][K_G] \right) [C] = 0 \tag{52}$$

where

$$[A_c] = [A] - [B][K_G][C] \tag{53}$$

The solution of the Lyapunov equation can be obtained by solving eigenvalue problem of the closed-loop system matrix. However, it is unstable when the eigensystem is ill-conditioned. Since aeroservoelastic system dealt with in this study is ill-conditioned, Lyapunov solver, utilizing Schur decomposition, is used. This algorithm transforms the closed-loop system matrix to Schur form by orthogonal similarity transformation, computes the solution of the resulting triangular system and transforms this solution back.

Another purpose of the control system is to prevent performance degradation due to external disturbances such as gust. For the gust response reduction system, the root mean square (RMS) values of gust response for the different elastic modes can be calculated for performance comparison. The state covariance matrix [X] of the closed-loop system is the solution of a Lyapunov equation in the form.

$$[A_c][X] + [X][A_c]^T + \{B_w\}[Q_w]\{B_w\}^T = 0 \tag{54}$$

where $[Q_w]$ is the intensity matrix of the white noise. The square of the RMS of the system outputs(the modes of the system) is computed as follows.

$$\sigma_i^2 = \left[[C][X][C]^T\right]_{ii}, i = 1,2,.., Ns \tag{55}$$

4.6 Example : Gust Response Reduction System Design

A composite wing model with inboard/outboard control surfaces is used as an example model for aeroelastic control system design. The model has 25 degrees of sweep angle and a taper ratio and aspect ration of the wing are 0.8 and 2,48, respectively. The wing area is set to be 1800 in^2. The wing skin is composed of 14 symmetric composite layers, which are $[-90/+45/-45/-40/+45/-45/-90]_s$. Each layer has uniform thickness, 0.008, 0.01, 0.01, 0.104, 0.01, 0.01 and 0.008 inches, respectively. The inboard control surface is located at 15% of span. The spanwise size of the control surface is 25% of span (i.e., Y_i=15%, L_i=25% of the span). The outboard control surface is located at 45% of span.

The spanwise size of the control surface is 30% of span (i.e., Y_o=45%, L_o=30% of the span). The chordwise size of the each control surface is set to be 25% of chord.

The actuator dynamics for each control surface is defined as

$$\frac{\{\delta_c\}}{\{\delta_{aileron}\}} = \frac{20}{s+20}\frac{1.6\times10^5}{s^2+400s+1.6\times10^5}$$

The modal reduction is conducted using first 6 vibration modes. The first 6 natural frequencies of the mode are 6.1, 15.0, 26.8, 38.7, 53.6, 72.3 Hz, respectively. For RFA using Roger's method, 4 aerodynamic poles are defined. These are 0.2, 0.4, 0.6, and 0.8. Therefore, the resulting aeroservoelatic state space model is 45th order including 6 elastic modes, 24 aerodynamic states, 6 actuator states and 3 gust states.

For this wing model, open loop flutter analysis is conducted. Flutter occurs at 1850 feet/sec(fps) and is due to the second mode (13.5 Hz).

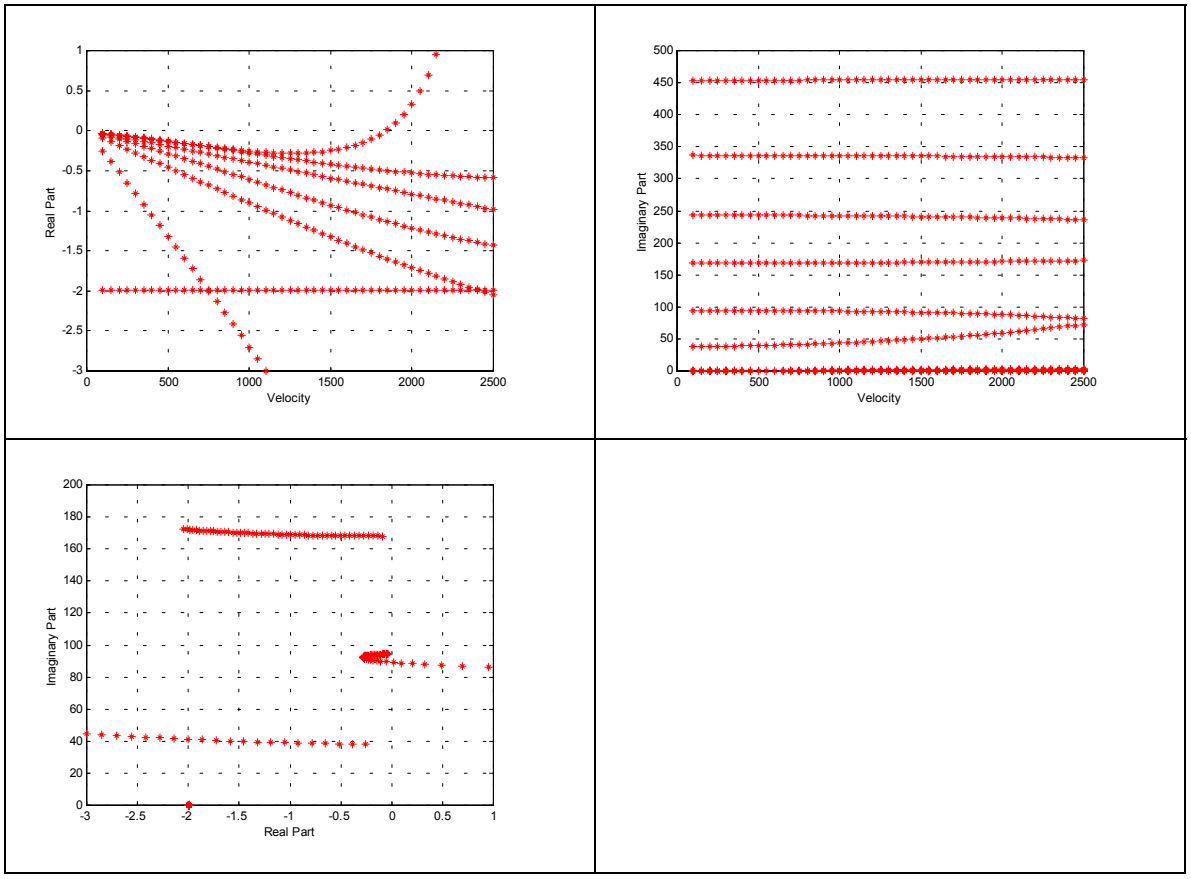


Figure Open Loop Flutter Analysis Result

The control system is designed at the design speed of 2000 fps. Table shows the open loop system eigenvalues at design speed. As shown in Table, 2nd mode becomes unstable. The desired eigenvalues are defined in order to design a control system using pole placement technique.

Table : Open Loop Eignvalues and Desired Closed Loop Eigen Values.

Open Loop Eigenvalues of Six Elastic Modes	Desired Closed Loop Eigenvalues
-6.5677e+000 +5.9379e+001i	-7.2245e+000 +5.9379e+001i
3.2996e-001 +8.7866e+001i	**-4.6195e-001 +8.7866e+001i**
-1.7126e+000 +1.7084e+002i	-1.7126e+000 +1.7084e+002i
-1.2166e+000 +2.3888e+002i	-1.2166e+000 +2.3888e+002i
-5.2780e-001 +3.3429e+002i	-5.2780e-001 +3.3429e+002i
-7.8903e-001 +4.5397e+002i	-7.8903e-001 +4.5397e+002i

The gain matrix is determined as

```
Columns 1 through 6
7.6267e-003 -4.6367e-003 -8.8729e-003 -5.0137e-002  5.8575e-004  1.0342e-003
3.1175e-002 -2.4089e-002 -4.6352e-003 -3.8585e-002  1.2787e-003 -5.7431e-005

Columns 7 through 12
 8.1452e-002  8.0865e-002 -1.3649e-002 -3.6692e-002 -2.3506e-003  1.2538e-003
-5.2469e-002 -7.4833e-002  2.1801e-002  5.8818e-002  1.3415e-002 -5.4627e-005
```

The RMS value of the gust response for the flexible modes can be used as a metric to compare the control performance.

List of Test.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%
clear all
close all

% NOTE : If you want to change one of the following,
%   you need to change input data for unsteady aero code.
%   check unsdlm.dat
 swep  =25.0; taper =0.8; aspct =2.48;
 y1bar =0.15; y2bar =0.6;
```

```
 l1bar =0.25; l2bar =0.3;
 ftc1=0.25;   ftc2=0.25;
%*wing area
 warea=1800.0;
% tc = wing thickness to chord (constant across span)
 tc=0.38;
% nele : number of elements
 nele=10;

 save wingdvs swep taper aspct y1bar y2bar l1bar l2bar ftc1 ftc2
%
% composite laminated plate
% nl is the number of layer
 nl=14;
%
 layerdata =[ 1.0000    0.9920  -90.0000    0.0080
              2.0000    0.9820   45.0000    0.0100
              3.0000    0.9720  -45.0000    0.0100
              4.0000    0.8680  -40.0000    0.1040
              5.0000    0.8580  -45.0000    0.0100
              6.0000    0.8480   45.0000    0.0100
              7.0000    0.8400  -90.0000    0.0800
              8.0000   -0.8480  -90.0000    0.0080
              9.0000   -0.8580   45.0000    0.0100
             10.0000   -0.8680  -45.0000    0.0100
             11.0000   -0.9720  -40.0000    0.1040
             12.0000   -0.9820  -45.0000    0.0100
             13.0000   -0.9920   45.0000    0.0100
             14.0000   -1.0000  -90.0000    0.0800    ];
%
% wti= non-structural weight of the wing per unit span
% wti is unit of slugs/in
 wti=[0.01345
      0.01111
      0.01099
      0.01172
      0.01134
      0.01051
      0.00681
      0.00502
      0.00502
      0.00502];
%
save dataprop nl layerdata wti
%
%
 getinput
%
% create the mass and stiffness matrices
%
 model
%
% solve eigenvalue problem
%
 modes
%
%  ================================
```

```
%  number of modes in reduced model
%  ================================
%
nmodes=6;
rmass =eivec(:,1:nmodes)'*gmass *eivec(:,1:nmodes);
rstiff=eivec(:,1:nmodes)'*gstiff*eivec(:,1:nmodes);
%
%
% calculate the unsteady aerodynamic forces
%
disp('Reading Unsteady Aerodynamic Forces ...')
iopen=input('Do you need to run aero code ? (y/n)=(1/0)   ');
%
if (iopen == 1),
!unsdlm
!prepost
end
%
%
% gust mode
disp('Let us include the gust mode.  please type 1')
igust=input('Do you want to include gust ? (y/n)=(1/0)   ');
%
%   ictrl : no. of control surfaces
need=nmodes;
save tranm eivec nele yb1 yb2 xroot yroot sweep
disp('Converting aerodynamic force matrix in modal coordinate....')
[ictrl,jnk,b0,rka,raero] = aero(igust,ndof,need) ;

if(igust==0),
need1=need+ictrl;
elseif(igust==1),
need1=need+ictrl+1;
end
%
% rational function approximation: roger's method
%
 xxx(1)=0.2;xxx(2)=0.4;xxx(3)=0.6;xxx(4)=0.8;
%
disp( 'writing rfa results .....')
[p0,p1,p2,p3,p4,p5,p6]=roger(igust,ictrl,need,jnk,xxx,rka,raero);
disp( 'rfa.==>done, assemble system matrix...')
%
% icls :0 for open loop , 1 for closed loop analysis
disp('icls :0 for open loop , 1 for closed loop analysis ')
icls=input('Do  you  want  to  design  closed  loop  system  ?  (y/n)=(1/0)
');
%
%
% i0vel : initial velocity (ft/sec)
% j0vel : final velocity (ft/sec)
% istep : increament in velocity
%
  if (icls == 0 ),
% open loop analysis
  i0vel=100; j0vel=2500; istep=50;
  elseif (icls == 1 ),
```

```
% set design airspeed
  ivel=input('set design airspeed = (2000)');
  i0vel=ivel; j0vel=ivel; istep=100;
  end
%
%
% low pass filter for gust mode
% input chlng,afrq,sigwg
% chlng : characteristic length(inch)
% afrq (rad/sec) : a of the low pass filter a/(s+a)
% sigwg : sigma_wg
%
 if(igust ~= 0),
 sigwg=1;   chlng=50;   afrq=160;
 end
%  input actuator dynamics
%  'numerator',xnctrln(i)
%  'denominator'
%
if(ictrl ~= 0),
 for i=1:ictrl
 xnctrln(i)=32.0e06;
 dnctrl3(i)=1; dnctrl2(i)=420; dnctrl1(i)=16.08e6; dnctrl0(i)=32.0e6;
 end
disp('actuator dynamics')
[xnctrln']
[dnctrl3' dnctrl2' dnctrl1' dnctrl0']
end

disp( 'Open Loop System Analysis ...')
system

if (icls ==1),
save openloop ivel a_sys b_sys bw c_sys eivec gsystem
end

figure(1)
plot(solopn(:,1),solopn(:,2),'*r'),grid on,axis([0 2500 -3 1]),
xlabel('Velocity'),ylabel('Real Part')

figure(2)
plot(solopn(:,1),solopn(:,3),'*r'),grid on,axis([0 2500 0 500]),
xlabel('Velocity'),ylabel('Imaginary Part')

figure(3)
plot(solopn(:,2),solopn(:,3),'*r'),grid on,axis([-3 1 0 300]),
xlabel('Real Part'),ylabel('Imaginary Part')

if (icls ==1),
disp('open loop system data is creadted at ivel !')
disp('If you want to design a control system, just run clsdsn !')
end
```

List of Test.m

```
% Pole placement technique;
% CHeck eigsrt.m file and main file before you design a control system
!!

load openloop ivel a_sys b_sys bw c_sys eivec gsystem


A=a_sys;B=b_sys;C=c_sys;
n=max(size(A));
% number of dsesired mode = elastic modes
neigd=6;
eigd=zeros(n,1);
[eigop,phi0,psi0]=eignsort(A);
[eigop,phi0,psi0]=select(eigop,phi0,psi0,neigd);
%
% set desired closed loop eigenvalues
%
for i=1:neigd;
 eigd(2*i-1)=-abs(real(eigop(2*i-1)))+sqrt(-1)*imag(eigop(2*i-1));
 if(i==1);
 eigd(2*i-1)=-1.1*abs(real(eigop(2*i-1)))+sqrt(-1)*imag(eigop(2*i-1));
 end;
 if(i==2);
 eigd(2*i-1)=-1.4*abs(real(eigop(2*i-1)))+sqrt(-1)*imag(eigop(2*i-1));
 end;
 eigd(2*i)=conj(eigd(2*i-1));
end
%
% design  controller using nonlinear programming

G=polenlps(A,B,C,eigd,neigd);

[eigcl,phicl,psicl]=eignsort(A-B*G*C);
[eigcl,phicl,psicl]=select(eigcl,phicl,psicl,neigd);
%disp('open loop eigenvalues')
%disp([eigop(1:2*neigd)])
disp('desired and closed eigenvalues')
disp([eigd(1:2*neigd) eigcl(1:2*neigd)])

%[eigop eigcl]
disp('gain matrix is')
disp([G])

% Closed Loop System at Design Airspeed
Ac=A-B*G*C;

disp('open loop/closed loop eigenvalues')
[eignsort(A) eignsort(Ac)]

sigma =rms(A ,B,C,bw);
sigmac=rms(Ac,B,C,bw);
disp('RMS values of the open/closed loop system due to disturbance')
[sigma sigmac]
```

Chapter 5
Aeroservoelastic Design of Composite Plate-Wing Using Piezoelectric Materials

To prevent the static and dynamic instability in a composite wing, we can passively or actively suppress the instability by a proper choice of ply orientation and actuating the control surface. Recently, the application of piezoelectric materials to structural control may provide a new dimension in design to eliminate the possibility of instability by changing wing configuration to cause lift distribution variation. There has been a considerable amount of research activity in using piezo sensor/actuators for the vibration control of flexible structures. Since the piezoelectric materials exhibit elastic deformation in proportion to the magnitude of an applied electric field and transfer forces to the structure, piezoelectric materials that are bonded at proper locations of a base structure can be used as actuators.

An analytic model for an induced strain actuator coupled with beams and plates has been developed by many authors. The static and dynamic aeroelastic behavior of a wing structure with piezo actuators has been studied. Ehlers and Weisshaar showed the effects of the piezoelectric actuators on the static aeroelastic behavior such as lift effectiveness, divergence, roll effectiveness. Lazarus, Crawley and Bohlmann used the typical section to represent a wing model and performed an analytic study to find the optimal placement and thickness of a piezo actuator for camber or twist control. Lazarus, Crawley and Lin showed the ability of both articulated control surfaces and piezoelectric actuators to control the dynamic aeroelastic systems. Heeg conducted an experimental study for flutter suppression of a beam model and demonstrated the application of piezoelectric materials for flutter suppression, which is affixed to a flexible mount system.

In this chapter, an aeroservoelastic system will be designed using the piezoelectric materials as actuators. The purpose of this chapter is to design an active control system for flutter suppression of a composite plate-wing with segmented piezoelectric actuators. The analysis of the laminated composite wing with segmented piezoelectric actuators is conducted by Rayleigh-Ritz method. The active control system design for flutter

suppression requires the equations of motion to be expressed in a linear time-invariant state-space form. Therefore, it is necessary to approximate the unsteady aerodynamic forces in terms of rational functions of the Laplace variables. Minimum state method will be introduced to approximate the frequency domain unsteady aerodynamic forces as the transfer functions of the Laplace variable.

5.1 Modeling of Wing Structures with Piezoelectric Elements

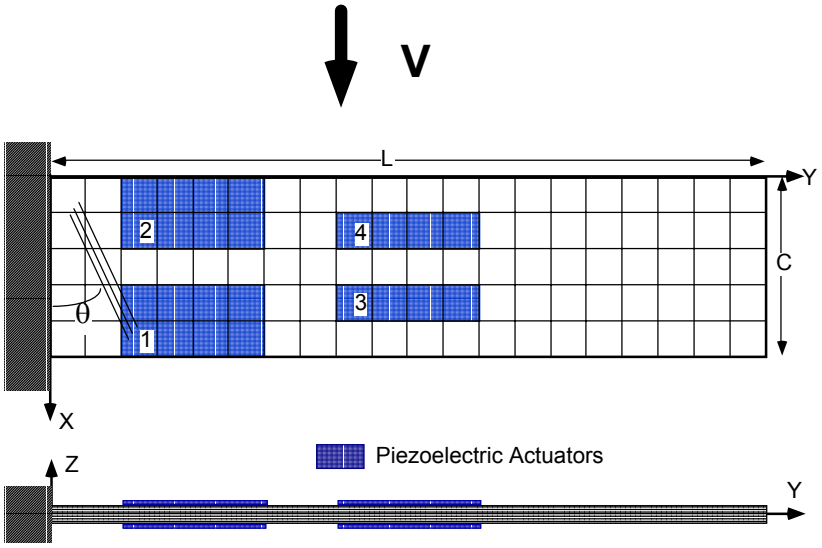


Figure Composite Plate-Wing with Pizeoelectric Actuatrors

Based on the classical laminated plate theory, the equations of motion of a model with piezo actuators are formulated. The linear coupled electromechanical constitutive relations of a piezoelectric materials can be written as;

$$\{D\} = [d]^T \{T\} + [\varepsilon^t]\{e\}$$
$$\{S\} = [s^e]\{T\} + [d]\{e\} \qquad (1)$$

{S} is the strain, {T} is the stress, {D} is the electric displacement, and {e} is the electric field intensity. [d],[ε^t],[s^e] denote the piezoelectric constant, dielectric constant, and elastic compliance matrix, respectively. It is assumed that the "3" axis is associated with the direction of poling and the mechanical property of the piezoelectric materials is modeled as an isotropic material. For the thin plate shape of the piezoelectric materials, we can rewrite Eq. (1).

$$D_3 = \begin{bmatrix} d_{31} & d_{32} & 0 \end{bmatrix} \begin{bmatrix} \sigma_1 & \sigma_2 & \sigma_{12} \end{bmatrix}^T + \varepsilon_{33}^t e_3$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} S_{p11} & S_{p12} & 0 \\ S_{p12} & S_{p22} & 0 \\ 0 & 0 & S_{p66} \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_{12} \end{Bmatrix} + \begin{Bmatrix} d_{31} \\ d_{32} \\ 0 \end{Bmatrix} e_3 \tag{2}$$

And, the stress-strain relations of a thin piezoelectric layer are

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} Q_{p11} & Q_{p12} & 0 \\ Q_{p12} & Q_{p22} & 0 \\ 0 & 0 & Q_{p66} \end{bmatrix} \left\{ \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{Bmatrix} - \begin{Bmatrix} d_{31} \\ d_{32} \\ 0 \end{Bmatrix} e_3 \right\} \tag{3.a}$$

or

$$\{\sigma\}_p = [Q_p]\{\{\varepsilon\} - \{\Lambda\}\} \tag{3.b}$$

$[Q_p]$ is the stiffness matrix of piezoelectric layer, $\{d_{ij}\}$ is the piezoelectric strain coefficient, and e_3 is the applied electric field. This equation is similar to stress-strain equation with thermal effects considering the fact that the piezoelectric strain Λ has the same form as thermal strain $\alpha\Delta T$.

Inplane forces and moments of laminated plate including the loads of the piezo actuators are obtained by integrating the stresses over the ply thickness, and can be expressed as follows,

$$\{N\} = \int \sigma dz = [A]\{\varepsilon\} + [B]\{\kappa\} - \{N_\Lambda\} \tag{4.a}$$

$$\{M\} = \int \sigma z dz = [B]\{\varepsilon\} + [D]\{\kappa\} - \{M_\Lambda\} \tag{4.b}$$

where ε, κ are the midplane strain and curvature, respectively. [A], [D], and [B] are extension, bending, and extension/bending coupling stiffness matrices, respectively. These matrices are influenced by not only the ply orientations of the laminate but also the

geometry of the piezo actuators. Inplane forces and moments due to actuator strain are given by

$$\{N_\Lambda\} = \int_{zp} [Q_p]\{\Lambda\} dz \tag{5}$$

$$\{M_\Lambda\} = \int_{zp} [Q_p]\{\Lambda\} z dz \tag{6}$$

where z_p is the coordinate through thickness in the piezoelectric material.

The strain energy of plate with piezoelectric material can be expressed as

$$U = \frac{1}{2} \iint_{A,A_p} \lfloor \varepsilon \kappa \rfloor \begin{bmatrix} A & B \\ B & D \end{bmatrix} \begin{Bmatrix} \varepsilon \\ \kappa \end{Bmatrix} dxdy - \iint_{A_p} \lfloor N_\Lambda M_\Lambda \rfloor \begin{Bmatrix} \varepsilon \\ \kappa \end{Bmatrix} dxdy \tag{7}$$

The kinetic energy is

$$T = \frac{1}{2} \iint_{A,A_p} \rho(\dot{u}^2 + \dot{v}^2 + \dot{w}^2) dxdy \tag{8}$$

where A, A_p represent the area of the composite plate and piezoelectric material, respectively, u,v,w are displacements in x, y, z direction. These energies are the functions of the ply orientations of the laminate and the piezoelectric parameters, which are the actuator thickness, area, and locations.

It is impossible to obtain closed form solutions to the laminated plate due to the complexity of equation, the arbitrary boundary conditions, and external forces such as unsteady aerodynamic forces. In this study, the Rayleigh-Ritz method which is faster and has the same accuracy as the finite element method is adopted for the structural analysis. To apply the Rayleigh-Ritz method, we introduce displacement functions in generalized coordinate system to represent displacements u, v, w

$$
\begin{aligned}
u(x,y,t) &= \sum_{i=1}^{l} X_i(x,y) q_i(t) \\
v(x,y,t) &= \sum_{j=l+1}^{l+m} Y_j(x,y) q_j(t) \\
w(x,y,t) &= \sum_{k=l+m+1}^{l+m+n} Z_k(x,y) q_k(t)
\end{aligned}
\tag{9}
$$

where X(x,y), Y(x,y), Z(x,y) are shape functions which must be properly chosen so as to satisfy the geometric boundary conditions of the wing structures. Using these displacement expressions, equations for strain energy and kinetic energy are written in matrix form,

$$
\begin{aligned}
U &= \frac{1}{2}\{q\}^T [K_s]\{q\} - \lfloor Q_\Lambda \rfloor \{q\} \\
T &= \frac{1}{2}\{\dot{q}\}^T [M_s]\{\dot{q}\}
\end{aligned}
\tag{10}
$$

where Q_Λ is the control force vector due to the actuation strain.

For the design of suppression system of a symmetric laminated plate model with piezoelectric actuators, the model is assumed to have the surface bonded piezoelectric actuators on the opposite side of the plate at the same location. It is also assumed that the same magnitude but the opposite direction of electric field is applied to the actuator in order to create a pure bending moment for flutter suppression. With these assumptions, the displacements in the x and y directions can be neglected. In order to express the displacement in the z direction, we used the free-free beam vibration modes in the x direction(chordwise direction) and the cantilever beam vibration modes in the y direction(spanwise direction) as follows;

$$
Z_k(x,y) = \Phi_i(x)\Psi_j(y) \tag{11}
$$

where

$$\Phi_i(x) = 1, i = 1$$

$$\Phi_i(x) = \sqrt{3}(1 - 2\frac{x}{c}), i = 2 \tag{12}$$

$$\Phi_i(x) = (\cos\alpha_i \frac{x}{c} + \cosh\alpha_i \frac{x}{c}) - \beta_i(\sin\alpha_i \frac{x}{c} + \sinh\alpha_i \frac{x}{c}), i = 3,4,5..$$

$$\Psi_j(y) = (-\cos\bar{\alpha}_j \frac{y}{l} + \cosh\bar{\alpha}_j \frac{y}{l}) - \beta_i(\sin\bar{\alpha}_j \frac{y}{l} + \sinh\bar{\alpha}_j \frac{y}{l}), \quad i = 1,2,3.. \tag{13}$$

In Eq.(10), if we express $\{Q_\Lambda\}$ in terms of the applied voltage {u}, it becomes as follows;

$$\begin{aligned}\{Q_\Lambda\} &= 2\sum_{i=1}^{np} \frac{1}{t_p}\{(\frac{t_s}{2} + t_p)^2 - (\frac{t_s}{2})^2\} \iint_{A_p} \left\{ [Q_p] \begin{Bmatrix} d_{31} \\ d_{32} \\ 0 \end{Bmatrix} \right\}^T \{Z''\} dxdy V(i) \\ &= [F_p]\{u\}\end{aligned} \tag{14}$$

where

$$\begin{aligned}\{Z''\}^T &= -\left\lfloor \frac{\partial^2 Z}{\partial x^2}, \frac{\partial^2 Z}{\partial y^2}, 2\frac{\partial^2 Z}{\partial x \partial y} \right\rfloor \\ \{u\}^T &= \lfloor V(1), V(2), ..., V(np) \rfloor\end{aligned} \tag{15}$$

t_s, t_p are the thicknesses of the laminate and piezoelectric layer, respectively. $[F_p]$ is the control force per unit voltage, np is the number of surface bonded piezoelectric actuators, V(i) is the applied voltage to the i-th piezoelectric actuator.

The virtual work due to aerodynamic forces can be rewritten as follow;

$$\delta W_A = \bar{q} \lfloor \delta w \rfloor_{c/4} [S]\{\Delta C_p\} = \bar{q} \lfloor \delta w \rfloor_{c/4} [S][AIC]^{-1} \{\bar{w}\}_{3c/4} \tag{16}$$

q is the dynamic pressure, a diagonal matrix [S] denotes the panel area, and ΔC_p is the pressure coefficient. And $\lfloor \delta w \rfloor_{c/4}$, $\{\overline{w}\}_{3c/4}$ denote the virtual displacement at 1/4 chord, the downwash at 3/4 chord of the panel, respectively. With harmonic motion assumption, the downwash is expressed as the following;

$$\{\overline{w}\}_{3c/4} = [\alpha_{ij}]_{3c/4}\{q\} - i\frac{\omega}{V}[W_{ij}]_{3c/4}\{q\} \tag{17}$$

where

$$\begin{aligned} [\alpha_{ij}]_{3c/4} &= -\left[\frac{\partial Z_j}{\partial x}\right]_{i(3c/4)} \\ [W_{ij}]_{3c/4} &= [Z_j]_{i(3c/4)} \end{aligned} \tag{18}$$

Z_j is the displacement function in z-direction and {q} is the generalized coordinate. Similarly, the virtual displacement is written as

$$\{\overline{w}\}_{c/4} = [Z_j]_{i(c/4)}\{q\}$$

Then, the virtual work becomes

$$\begin{aligned} \delta W_A &= \overline{q}\{\delta q\}\lfloor Z_j \rfloor^T_{i(c/4)}[S][AIC]^{-1}\left[[\alpha_{ij}]_{3c/4} - i\frac{\omega}{V}[W_{ij}]_{3c/4}\right]\{q\} \\ &= \overline{q}\{\delta q\}[A]\{q\} \end{aligned} \tag{19}$$

It has to be noted that the aerodynamic forces, [A] is function of the reduced frequency and Mach number.

Lagrange's equations result in a set of ordinary differential equations of motion as follows;

$$[M_s]\{\ddot{q}\} + [K_s]\{q\} = [F_p]\{u\} + \overline{q}[A]\{q\} \tag{20}$$

where $\bar{q}$ is the dynamic pressure, [A] is the unsteady aerodynamic force matrix. The aerodynamic force matrix can be calculated using modified strip theory or doublet-lattice method as discussed in the previous chapter. $[M_s]$ and $[K_s]$ are respectively the generalized mass and stiffness matrices including the effect of the piezoelectric actuators placement, $[F_p]$ is the control force matrix due to unit applied electric field.

After the vibration analysis, a model reduction can be performed to obtain a set of equations of motion in the modal coordinates using first rm vibration modes.

$$[\overline{M}_s]\{\ddot{\xi}\} + [\overline{K}_s]\{\xi\} = [\overline{F}_p]\{u\} + \bar{q}[\overline{A}]\{\xi\} \tag{21}$$

where

$$\begin{aligned} [\overline{M}_s] &= [\Phi]^T [M_s][\Phi] \\ [\overline{K}_s] &= [\Phi]^T [K_s][\Phi] \\ [\overline{A}] &= [\Phi]^T [A][\Phi] \\ [\overline{F}_p] &= [\Phi]^T [F_p] \end{aligned} \tag{22}$$

$[\Phi]$ is the eigen matrix.

5.2 Frequency Domain Unsteady Aerodynamics and Rational Function Approximation

The classical aeroelastic analysis such as the V-g method and p-k method is performed in the frequency domain using unsteady aerodynamic forces computed for simple harmonic motion. As discussed in the previous section, in frequency domain, unsteady aerodynamic forces can be obtained from the following equation by DLM.

$$\begin{aligned} \delta W_A &= \bar{q}\{\delta q\}\left\lfloor Z_j \right\rfloor_{i(c/4)}^T [S][AIC]^{-1}\left[[\alpha_{ij}]_{3c/4} - i\frac{\omega}{V}[W_{ij}]_{3c/4} \right]\{q\} \\ &= \bar{q}\{\delta q\}[A]\{q\} \end{aligned} \tag{23}$$

However, for the aeroservoelastic analysis and design, it is necessary to transform the equations of motion into the state space form. In this chapter, the Minimum State method will be introduced for the rational function approximation. It is known that in minimum state method, the increase in the size of the augmented aerodynamic state is smaller than any other methods.

Minimum State Method

Minimum State method approximates the aerodynamic force matrix by

$$[\overline{A}_{ap}(\overline{s})] = [\overline{A}_0] + [\overline{A}_1]\overline{s} + [\overline{A}_2]\overline{s}^2 + [D'][\overline{s}I - R']^{-1}[E]\overline{s} \tag{24}$$

where $\overline{s}$ is the nondimensionalized Laplace variable($\overline{s} = sb/V$). V is the airspeed, b is the semi chord, and s is the Laplace variable. The components of diagonal matrix [R'] are negative constants which are selected arbitrarily. For a given [R'] matrix, $[\overline{A}_0]$, $[\overline{A}_1]$, $[\overline{A}_2]$, [D'], and [E] are determined by using repeated least-square fit.

First, let's define the aerodynamic states, $\{x_a\}$

$$\overline{s}\{x_a\} = [E]\overline{s}\{\xi\} + [R']\{x_a\} \tag{25}$$

Therefore,

$$\{x_a\} = [\overline{s}[I] - [R']]^{-1}[E]\overline{s}\{\xi\} \tag{26}$$

Then, aeroservoelastic equation will be

$$\begin{aligned}[\overline{M}_s]\{\ddot{\xi}\} + [\overline{C}_s]\{\dot{\xi}\} + [\overline{K}_s]\{\xi\} &= \overline{q}[[\overline{A}_0] + [\overline{A}_1]\overline{s} + [\overline{A}_2]\overline{s}^2]\{\xi\} \\ &\quad + \overline{q}[D'][\overline{s}[I] - [R']]^{-1}[E]\overline{s}\{\xi\} + [\overline{F}_p]\{u\}\end{aligned} \tag{27}$$

$$[\overline{M}_s]\{\ddot{\xi}\} + [\overline{C}_s]\{\dot{\xi}\} + [\overline{K}_s]\{\xi\} = \overline{q}[[\overline{A}_0] + [\overline{A}_1]\overline{s} + [\overline{A}_2]\overline{s}^2]\{\xi\} + \overline{q}[D']\{x_a\} + [\overline{F}_p]\{u\}$$

After rearranging the equation, the system equations can be expressed

$$\left[\overline{M}_s - \overline{q}\overline{A}_2\left(\frac{b}{V}\right)^2\right]\{\ddot{\xi}\} = -\left[\overline{C}_s - \overline{q}\overline{A}_1\left(\frac{b}{V}\right)\right]\{\dot{\xi}\} - \left[\overline{K}_s - \overline{q}\overline{A}_0\right]\{\xi\} + \overline{q}\left[D'\right]\{x_a\} + [\overline{F}_p]\{u\} \tag{28}$$

$$\left[\widetilde{\overline{M}}\right]\{\ddot{\xi}\} = -\left[\widetilde{\overline{C}}\right]\{\dot{\xi}\} - \left[\widetilde{\overline{K}}\right]\{\xi\} + \overline{q}\left[D'\right]\{x_a\} \tag{29}$$

where

$$\begin{aligned}\left[\widetilde{\overline{M}}\right] &= \left[\overline{M}_s - \overline{q}\overline{A}_2\left(\frac{b}{V}\right)^2\right] \\ \left[\widetilde{\overline{C}}\right] &= \left[\overline{C}_s - \overline{q}\overline{A}_1\left(\frac{b}{V}\right)\right] \\ \left[\widetilde{\overline{K}}\right] &= \left[\overline{K}_s - \overline{q}\overline{A}_0\right]\end{aligned} \tag{30}$$

Since $\overline{s}\{x_a\} = \left[E\right]\overline{s}\{\xi\} + \left[R'\right]\{x_a\}$, the aerodynamic state is expressed as

$$\{\dot{x}_a\} = \left[E\right]\{\dot{\xi}\} + \left[R'\right]\left(\frac{V}{b}\right)\{x_a\} \tag{31}$$

Using the state vector $\{x\}$, the linear time-invariant state space equations of motion which include the effects of piezoelectric control forces are expressed as follows

$$\{\dot{x}\} = [F]\{x\} + [G]\{u\} \tag{32}$$

where

$$[F] = \begin{bmatrix} 0 & [I] & 0 \\ -\left[\widetilde{\overline{M}}\right]^{-1}\left[\widetilde{\overline{K}}\right] & -\left[\widetilde{\overline{M}}\right]^{-1}\left[\widetilde{\overline{C}}\right] & q_d\left[\widetilde{\overline{M}}\right]^{-1}[D'] \\ 0 & [E] & \left[R'\right]\left(\frac{V}{b}\right) \end{bmatrix}$$

$$[G] = \begin{bmatrix} 0 \\ \left[\widetilde{\overline{M}}\right]^{-1}\left[\overline{F}_p\right] \\ 0 \end{bmatrix}$$

The state vector, {x} is defined as $\{x\} = \begin{bmatrix} \xi & \dot{\xi} & x_a \end{bmatrix}^T$. If [R'] is set to be a (m × m) matrix, the total number of states is (2*rm × m). It should be noted that the system matrix [F] if function of airspeed.

5.3 Optimal Output Feedback Design

Since the system matrix is a function of the dynamic pressure, it is necessary to choose the design air speed for designing control system. By choosing the design air speed(V_{Design}), the linear quadratic regulator(LQR) with output feedback is applied for design of the flutter suppression system. The design speed, which is chosen arbitrarily is usually larger than the open loop flutter speed.

The design procedure for LQR with output feedback is briefly summarized below.

Consider a linear time-invariant system

$$\begin{aligned} \{\dot{x}\} &= [F]\{x\} + [G]\{u\} \\ \{y\} &= [H]\{x\} \end{aligned} \tag{33}$$

The output matrix [H] is specified such that the aerodynamic states are excluded from the state for feedback purpose. The admissible controls are output feedback of the form

$$\{u\} = -[K_G]\{y\} \tag{34}$$

where $[K_G]$ is an output feedback gain matrix, which consists of constant coefficients to be determined by the design procedure. The performance index to be minimized is selected as follows:

$$J = \int_0^\infty [\{y\}^T [Q]\{y\} + \{u\}^T [R]\{u\}]dt \tag{35}$$

where $[Q] \geq 0$ and $[R] > 0$ are symmetric weighting matrices. Since the initial state $\{x_0\}$ may not be known, we adopt Athans and Levine's procedure that minimizes the expected value of performance index

$$\bar{J} = E\{J\} = trace\{[P][X_0]\} \tag{36}$$

where the operator $E\{\bullet\}$ denotes the expected value over all possible initial conditions. The symmetric matrices [P] and $[X_0]$ are defined by

$$[X_0] = \{x_0\}\{x_0\}^T \tag{37}$$

$$[P] = \int_0^\infty e^{[F_c]^T t}[H]^T \left\{[Q] + [K_G]^T [R][K_G]\right\}[H]e^{[F_c]t}dt \tag{38}$$

where

$$[F_c] = [F] - [G][K_G][H] \tag{39}$$

The usual assumption is that the initial states are uniformly distributed on the unit sphere; then $[X_0]$ becomes identity matrix. It is well known that matrix [P] can be obtained by solving the following Lyapunov equation

$$[\Sigma] = [F_c]^T [P] + [P][F_c] + [H]^T [Q][H] + [H]^T [K_G]^T [R][K_G][H] = [0] \tag{40}$$

Note that Lyapunov equation is a symmetric linear matrix equation. The solution of the Lyapunov equation can be obtained directly by solving simultaneous linear equations, however this approach takes lots of computation time and storage. One of the most

efficient and robust algorithms to solve Lyapunov equations uses QR factorization. In this algorithm, the Lyapunov equation is transformed into a simpler equation by use of Schur transformations. Now, our problem is to determine the optimal output feedback gain matrix [K_G] at the design speed, by minimizing the performance index subject to the algebraic matrix constraint Eq.(40).

To solve this problem with a constraint equation, a Lagrange multiplier approach is widely used. The Hamiltonian is defined by adjoining the constraint Eq.(40) to the performance index and expressed as follows.

$$\Re = trace\{[P][X_0]\} + trace\{[\Sigma][S]\} \tag{41}$$

[S] is a symmetric matrix of Lagrange multipliers which still needs to be determined. The following conditions are necessary for [P], [S], and [K_G] to be the extrema:

$$\frac{\partial \Re}{\partial [P]} = [0], \frac{\partial \Re}{\partial [S]} = [0], \frac{\partial \Re}{\partial [K_G]} = [0] \tag{42}$$

Applying Eq.(42) to Eq.(41), necessary conditions for optimality are obtained as follows:

$$\begin{aligned} &[F_c]^T[P] + [P][F_c] + [H]^T[Q][H] + [H]^T[K_G]^T[R][K_G][H] = [0] \\ &[F_c][S] + [S][F_c]^T + [X_0] = [0] \\ &[R][K_G][H][S][H]^T - [G]^T[P][S][H]^T = [0] \end{aligned} \tag{43}$$

There exist several numerical approaches to solve Eqs. (43). Gradient-based routines can be used with various optimization algorithms. Moerder and Calise proposed an efficient iterative algorithm, which converges to a local minimum if the following conditions hold:

(i) there exists an output gain matrix [K_G] such that [F_c] is stable,

(ii) the output matrix [H] has full row rank,

(iii) control weighting matrix [R] is positive definite, and

(iv) output state weighting matrix [Q] is positive semidefinite and should be chosen so that all unstable states are weighted in the performance index.

The iterative algorithm for finding solution is outlined as follows:

Step 1: Choose $[K_G]$ such that $[F_c]$ in Eq.(39) has all the eigenvalues with negative real parts.

Step 2: With given $[K_G]$ and the corresponding $[F_c]$, solve the Lyapunov Eqs.(43) for [P] and [S], and compute current performance index value.

Step 3: Evaluate the gain update direction

$$\Delta[K_G] = [R]^{-1}[G]^T[P][S][H]^T([H][S][H]^T)^{-1} - [K_G] \tag{44}$$

Step 4: Update the gain by

$$[K_G]_{new} = [K_G] + \alpha\Delta[K_G] \tag{45}$$

where α is chosen so that the closed-loop system $[F_c]$ is asymptotically stable, and the performance index of the updated system is less than that of the current system. α is usually set to be 0.3.

Step 5: Iterate Steps 2 through 4 until the converged solution is obtained.

Any minimization algorithm requires the selection of an initial stabilizing gain. However, such a gain may or may not be easy to find. One possible way to find the initial gain matrix is to use root locus techniques by closing one loop at a time in the control system. Here, first we can solve the full state LQR problem with performance index Eq.(35), and construct initial stabilizing output gain matrix by choosing the

corresponding gain parameters to be used in the output feedback from the resulting LQR gain matrix elements.

5.4 Aeroservoelastic Design Example

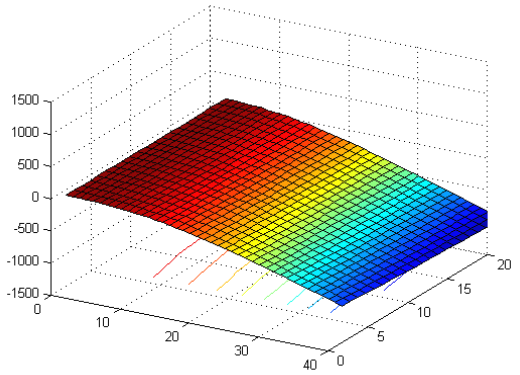
The plate model containing four sets of piezoelectric actuators is used as a model for the control system design. It is assumed that a set of actuators is bonded on both top and bottom surfaces of the laminated plate in order to generate a pure bending force. The size of each piezo actuator is 1.5×3.0 inch. The material properties used in this study are given in Table 1.

Table 1. Material Properties

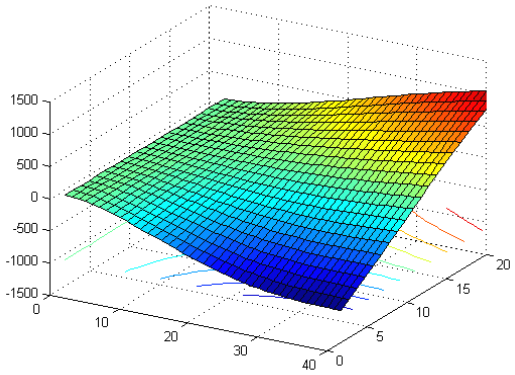
Composite Materials	Piezoelectric Materials
$E_1=14.21\times10^6$ psi	$E_p=9.137\times10^6$ psi
$E_2=1.146\times10^6$ psi	$\rho_p=0.28$ lb/in^3
$G_{12}=0.8122\times10^6$ psi	$\nu_p=0.3$
$\rho=0.05491$ lb/in^3	$d_{31}=6.5\times10^{-9}$ in/V
$\nu_{12}=0.28$	$d_{32}=6.5\times10^{-9}$ in/V
	$\nu_{12}=0.28$

The laminate has six symmetric layers, $[105/\pm45]_s$. Each layer has uniform thickness, which is 0.02 inch, respectively. The coordinates of the piezo actuators are pzx(i)= {0.50, 0.50, 3.50, 3.50} inch and pzy(i)={1.00, 7.00, 1.00, 7.00} inch.

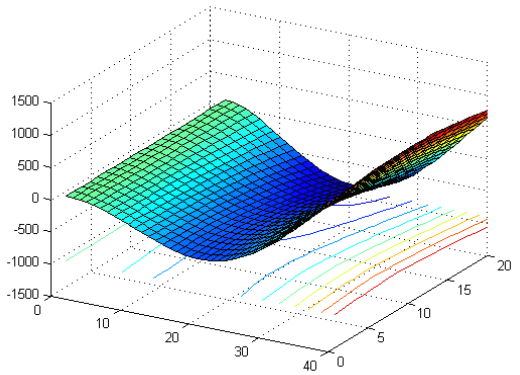
After vibration analysis, first six modes are used for the modal reduction. The first six natural frequencies are 34.9, 122.3, 220.8, 415.3, 523.5, 624.1 Hz, respectively. Figure ?? shows the first mode shapes used for the modal reduction.



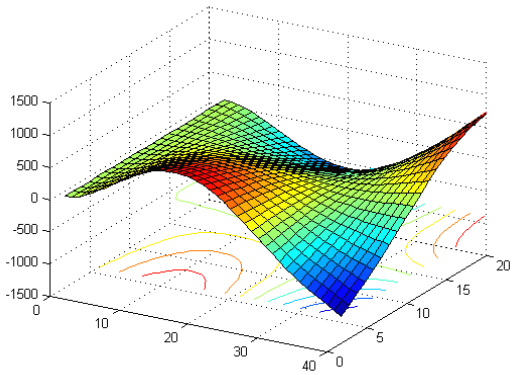
1st mode



2nd mode



3rd mode



4th mode

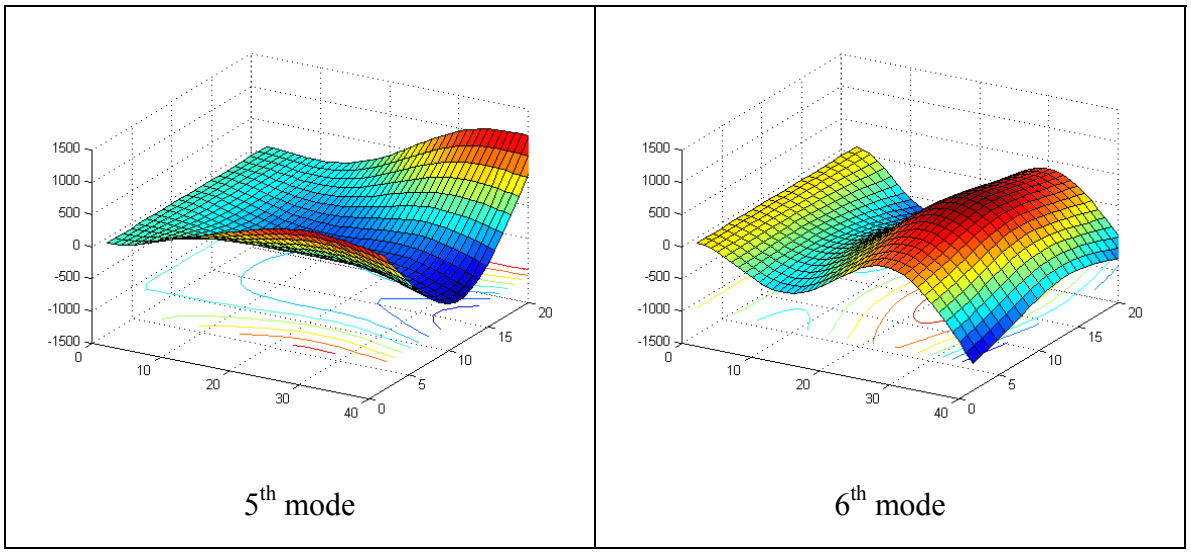
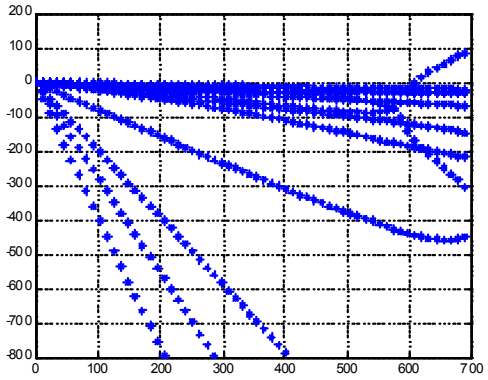


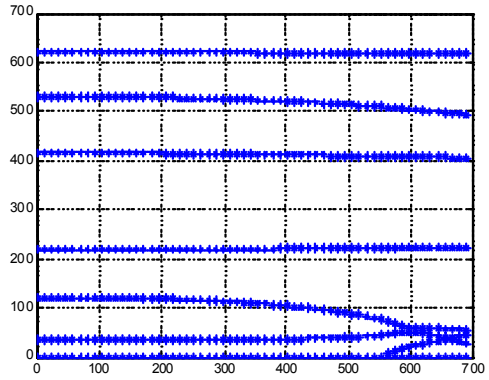
Figure Vibration mode shapes of the laminated Composite Plate-Wing with Piezoelectric Actuators.

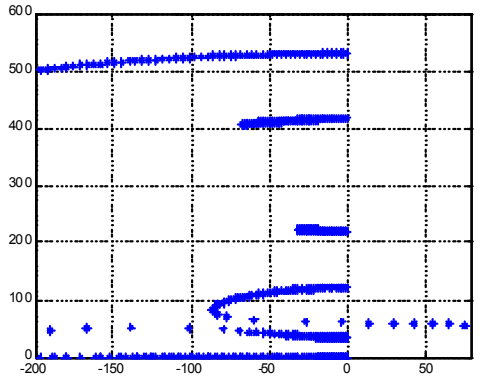
Unsteady aerodynamics are calculated using DLM for Mach 0.8. For calculation of the pressure distribution on an oscillating plate-wing undergoing simple harmonic motion, the plate-wing model is uniformly divided into total 100 panels arranged in 10 spanwise direction and 10 chordwise direction.

The open loop flutter analysis of $[105/\pm45]_s$ laminate is conducted with the unsteady aerodynamics transformed into Laplace domain through Minimum State method. The result is shown in Figure. For the rational function approximation, a total of 6 components of the [R'] matrix are used for the RFA. Therefore, the dimension of the state vector is 18.

The diagonal components of [R'] matrix are {-0.0072, -0.048, -0.197, -0.49, -0.67, -0.98}, respectively. As in Figure, in $[105/\pm45]_s$ model without a control system, flutter occurs at 620 fps by the 2nd mode, which is the 1st torsion mode dominant and the flute frequency is about 59 Hz.







Open Loop Analysis Results

$[105/\pm 45]_s$ laminated composite with 4 PZT actuators.

Flutter speed = 620 ft/sec

Flutter frequency = 59 Hz

Figure Open Loop Flutter Analysis Result of $[105/\pm 45]_s$ Composite Plate-Wing with Piezoelectric Actuators.

For the active control design, the design speed is set to be 800 fps, which is chosen arbitrarily. At the design speed, the open loop system is unstable in the first torsion mode, yielding unstable eigenvalues at (162.9 ± I 294.2). The open loop matrix equation is supplemented with control term. The control input {u} is determined by LQR with output feedback subject to minimize the performance index at the design speed. In order to select the initial gain matrix, first we solve the full state LQR problem with performance index, and construct initial stabilizing output gain matrix by choosing the corresponding gain parameters to be used in the output feedback from the resulting LQR

gain matrix elements. For the full state LQR problem, the weighting matrices [Q] and [R] in Eq.(31) are set to be 500[I] and [I], respectively.

The loci of the closed loop aeroelastic roots with increasing airspeed is plotted in Figure. Above the design speed, the control law provides flutter mode control. The controller push the unstable eigenalvues to (-124.8 ± I 315.5). And the model is stable until a divergence occurs at about 900 fps.

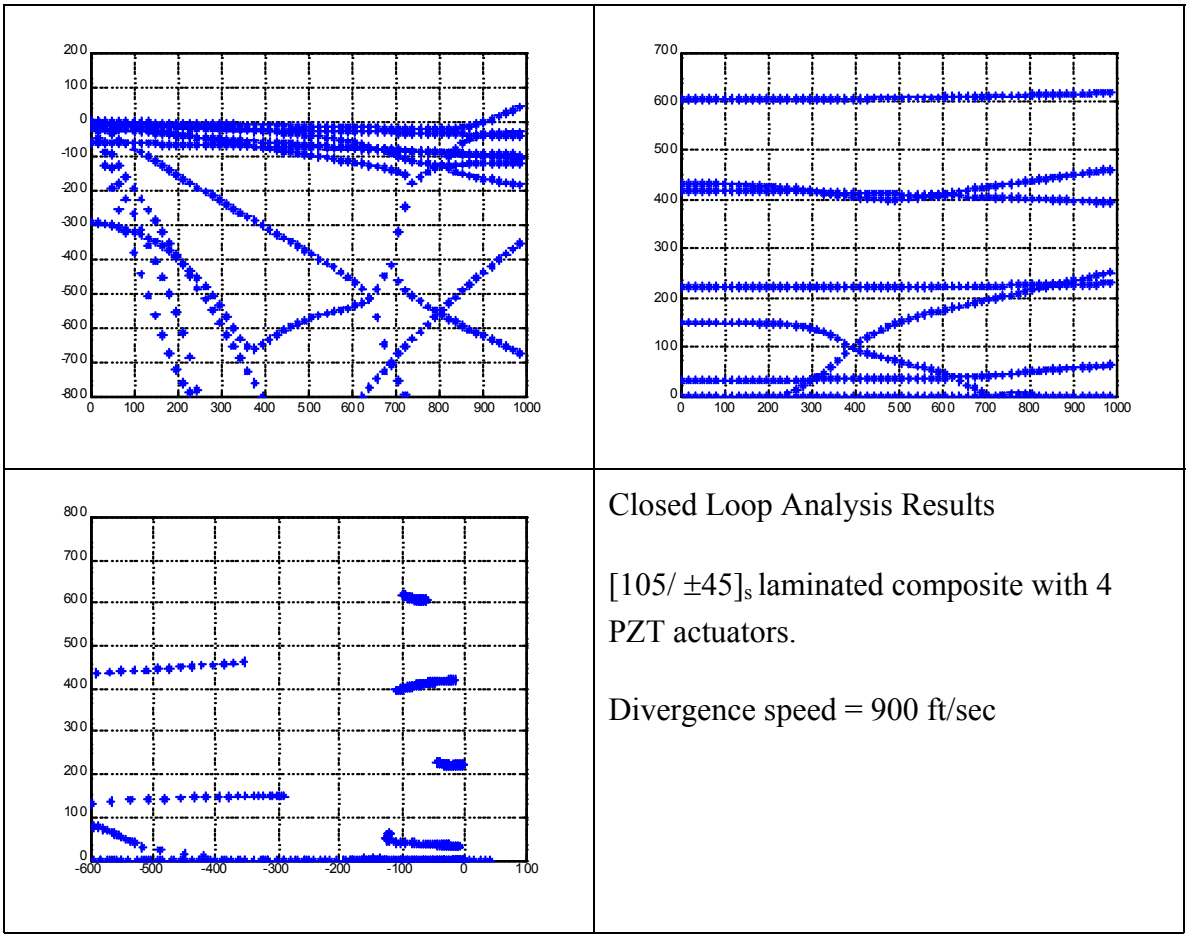


Figure Closed Loop Flutter Analysis Result of $[105/\pm45]_s$ Composite Plate-Wing with Piezoelectric Actuators.

5.5 Further Parametric Study

In this chapter, a control system is designed is to suppress the flutter of a composite wing using segmented piezoelectric actuators. The control system design, especially using optimal output feedback design, is performed for the active flutter suppression. An example model shows that the PZT actuators are capable of aeroelastic control. For the aeroelastic control, the piezoelectric actuator must create substantial changes in surface shape to generate the aerodynamic loads. The piezoelectric actuator location, size, and thickness should be considered in the control system design since these parameters may affect the structural properties as well as the control characteristics. The optimization technique can be applied to find the best geometry of actuators for flutter control subject to minimize the control performance index.

The closed loop damping ratios can be imposed in order that the control system has more stability margin, therefore higher critical aeroelastic airspeed. However, heavy constraints on the damping ratio may result in higher gain matrix and much control efforts required for flutter suppression. As the optimization progresses, a trade-off between the reduction of performance index and satisfaction of constraints on the closed damping ratios is needed in order to determine the best geometry of the piezo actuators. Also, the electric power requirement should be addressed.

References

1. Crawley, E.F., deLuis, J., ”Use of Piezo-Ceramics as Distributed Actuators in Large Space Structures,” AIAA Paper No. 85-0626, Proceedings of the 26th Structures, Structural Dynamics and Material Conference, NY, April 1985.

2. Crawley, E.F., deLuis, J., ”Use of Piezoelectric Actuators as Elements of Intelligent Structures,” AIAA Journal, Vol.25, No.10, Oct. 1987, pp.1373-1385.

3. Crawley, E.F., Anderson, E.H., ”Detailed Models of Piezoceramic Actuation of Beams,” AIAA Paper No. 89-1388, Proceedings of the 30th Structures, Structural Dynamics and Material Conference, AL, April 1989.

4. Ehlers, S.M., Weisshaar, T.A., ”Static Aeroelastic Behavior of an Adaptive Laminated Piezoelectric Composite Wing,” AIAA Paper No. 90-1078, Proceedings of the 31th Structures, Structural Dynamics and Material Conference, CA, April 1990.

5. Ehlers, S.M., Weisshaar, T.A., ”Effects of Adaptive Material Properties on Static Aeroelastic Control,” AIAA Paper No. 92-2526, Proceedings of the 33th Structures, Structural Dynamics and Material Conference, TX, April 1992.

6. Lazarus, K.B., Crawley, E.F., Lin, C.Y., ”Fundamental Mechanisms of Aeroelastic Control with Control Surface and Strain Actuation,” AIAA Paper No. 91-0985, Proceedings of the 32th Structures, Structural Dynamics and Material Conference, MD, April 1991.

7. Heeg, J., ” An Analytical and Experimental Investigation of Flutter Suppression Via Piezoelectric Actuators,” AIAA Paper No. 92-2106, Proceedings of the AIAA Dynamics Specialist Conference, April 1992.

7. Jones, R.M. , Mechanics of Composite Materials, Scripta Book Co., Washington, D.C.,1975.

8. Whitney, J.M., Structural Analysis of Laminated Anisotropic Plates, Technomic Publishing Co. INC., Lancaster, Pennsylvania, 1987.

9. Albano, E., Rodden, W.P., ”A Doublet-Lattice Method for Calculating Lift Distributions on Oscillating Surfaces in Subsonic Flows,” AIAA Journal, Vol. 7, No. 2, Feb. 1969, pp. 279-285.

10. Karpel, M., ”Design for Active Flutter Suppression and Gust Alleviation Using State-Space Aeroelastic Modeling,” Journal of Aircraft, Vol.19, No.3, March 1982, pp.221-227.

11. Hoadley, S.T., Karpel, M., ”Application of Aeroservoelastic Modeling Using Minimum-state Unsteady Aerodynamic Approximations,” Journal of Guidance, Control, and Dy-namics, Vol.14, No.6, November-December 1991, pp. 1267-1276.

12. Levine, W.S., and Athans, M., ”On the Determination of the Optimal Constant Output Feedback Gains for Linear Multivariable Systems,” IEEE Transactions on Automatic Control, Vol. AC-15, No. 1, February 1970, pp. 44-48.

13. Moerder, D.D., and Calise, A.J., ”Convergence of a Numerical Algorithm for Calculating Optimal Output Feedback Gains,” IEEE Transactions on Automatic Control, Vol. AC-30, No. 9, September 1985, pp. 900-903.

14. Lewis, F.L., Applied Optimal Control and Estimation, Prentice-Hall, New Jersey, 1992.

15. Mueller, G.S., and Adeniyi, V.O., ”Optimal Output Feedback by Gradient Methods with Optimal Stepsize Adjustment,” Proceedings of IEE, Control & Science, Vol. 126, No. 10, October 1979.

16. Kailath, T., Linear Systems, Prentice-Hall, New Jersey, 1980.

17. Chen, C.F., and Shieh, L.S., ”A Note on Expanding PA + A T P = Q,” IEEE Transactions on Automatic Control, Vol. AC-13, No. 2, February 1968, pp. 122-123.

18. Bartels, R.H., and Stewart, G.W., ”Algorithm 432: Solution of Matrix Equation AX + XB = C,” Comm. of the ACM, Vol. 15, No. 9, September 1972, pp. 820-826.

19. Kirk, D.E., Optimal Control Theory, Prentice-Hall, New Jersey, 1970.

