Assignment 7: Functions and Numerical Integration

Selected Solutions Instructor: Trani

Problem 1

A civil engineer is designing a rainstorm water management system for a shopping mall. During a severe thunderstorm, the water runoff generated by the large parking lot at the shopping mall is given by the function:

$$runoff = k_2 + k_1 \sin(t / k_3) e^{(-t/k_4)}$$

Where *runoff* is the runoff volume (cubic meters per second) generated by the parking lot, *t* is the time (in seconds) after the thunderstorm starts and k_1 through k_4 are parameters of the runoff function.

Task 1

Create a Matlab function to calculate the runoff for a given value of time t. As part of the input variables to the function runoff, include the four input parameters k1 through k4 for a 100 year storm with numerical values as follows:

```
k1 = 65;
k2 = 2;
k3 = 750;
k4 = 400;
```

Task 2

Create a Matlab script to call the function created in Task 1. Test the function for values of time (t) starting at t=0 and ending at t=3000 seconds. Plot the runoff as a function of time. Label appropriately.

Task 3

Modify the Matlab script created in Task 2 to find the area under the curve of the runoff as a function of time. Use the Matlab "quad" function to estimate the integral. Plot the results and output the result of the area under the curve in the command window using the "disp" function in Matlab. How much runoff is produced in the 3000 second storm event? State the units of the answer.

Task 4

Estimate the dimensions of a ponding volume needed to store all the runoff volume generated by the 100 storm event. State dimensions of the ponding volume (base x width x height).

Problem 2

The Manning equation is a simple empirical relationship used by civil engineers to estimate the flow characteristics inside pipes. The basic Manning Equation is:

$$Q = [1.486\ A * R^{(2/3)} * S^{(1/2)}] / n$$

where:

Q is the discharge (in cubic feet per second)

R is the hydraulic radius (ft : area of section / wetted perimeter)

S is the slope of the pipe invert (ft/ft)

A is the cross sectional area of flow (ft-ft)

n is the pipe roughness coefficient (see table below)

Table 2. Typical pipe roughness coefficients.

Type of Pipe	Roughness Coefficient
Concrete and asbestos	0.012
Corrugated metal	0.023

Task 1

Create a function to estimate Q in the Manning equation given all four parameters (A, R, S and n). In this function the fourth parameter (n) must be passed to the function as a string variable (i.e., either 'concrete' or 'metal') with the name of the pipe type. Inside your function detect the pipe type and assign the corresponding value of the pipe roughness coefficient (n). Test the function with the following values: S=0.2 ft/ft, R =15 inches and concrete pipes.

Task 2

Create a Matlab script and use the function created in Task 1 to generate solutions to the Manning equation for circular pipes (one for concrete and one for metal) flowing at full flow wetted areas (with radii ranging from 12-100 inches) and slopes ranging from 1 foot per 5 feet to 1 foot to 50 feet (0.2 to 0.02). Create a mesh plot in Matlab that shows the discharge (in z-axis), the slope (in y-axis) and the radius of the pipe in the x-axis for the ranges stated above.

Problem 3

A train engineer collects data during the certification of the new high-speed train to be introduced in the Northeast Corridor in the United States. The data collected records train acceleration (a) vs. velocity (V). The data is presented in the table below.

Table 1. Train Acceleration and Speed Data.

Train Velocity (m/s)	Maximum Train Acceleration (m/s^2)
0.00	2.1
20	1.56
30	1.30
40	1.06
50	0.76
60	0.51
80	0.00

Task 1

Create a Matlab script to plot the data and (in the same script) to find the best second order polynomials that fit the data (i.e., use the "polyfit" command). Save the coefficients of the polynomials and evaluate 100 data points of these polynomials across a range of speeds from 0-80 m/s (the usable operating speeds for this train). Compare with the original data.

```
% Script to estimate best curve fit for two vectors
clear
clc

% T. Trani

% Task 1

% Define two vectors for velocity and acceleration

velocity      = [0 20 30 40 50 60 80] ;                  % velocity in m/s
acceleration  = [2.1 1.56 1.30 1.06 0.76 0.51 0.00];    % accceleration (m/s-s)

% Do a  basic polynomial fit

coefficients = polyfit(velocity,acceleration,2);            % fits a second order polynomial

% Evaluate the polynomial found for the range of velocities of the train in
% the table

velNew = min(velocity):1:max(velocity);       % define a new velocity vector to evaluate the polynomial
accelerationFromPolyFit = polyval(coefficients,velNew);   % evaluate the polynomial using the coefficients found

% Make a plot and compare

% Create a label for the plot with the values of coefficidents found

labelPlot = horzcat('Acceleration = ', num2str(coefficients(1)), ' * Velocity^2 + ', ...
    num2str(coefficients(2)), ' * Velocity + ', num2str(coefficients(3)));

figure
plot(velocity,acceleration,'or',velNew,accelerationFromPolyFit,'b--')
xlabel('Train Velocity (m/s)','fontsize',20)
ylabel('Train Acceleration (m/s-s)','fontsize',20)
title(labelPlot)
grid
```

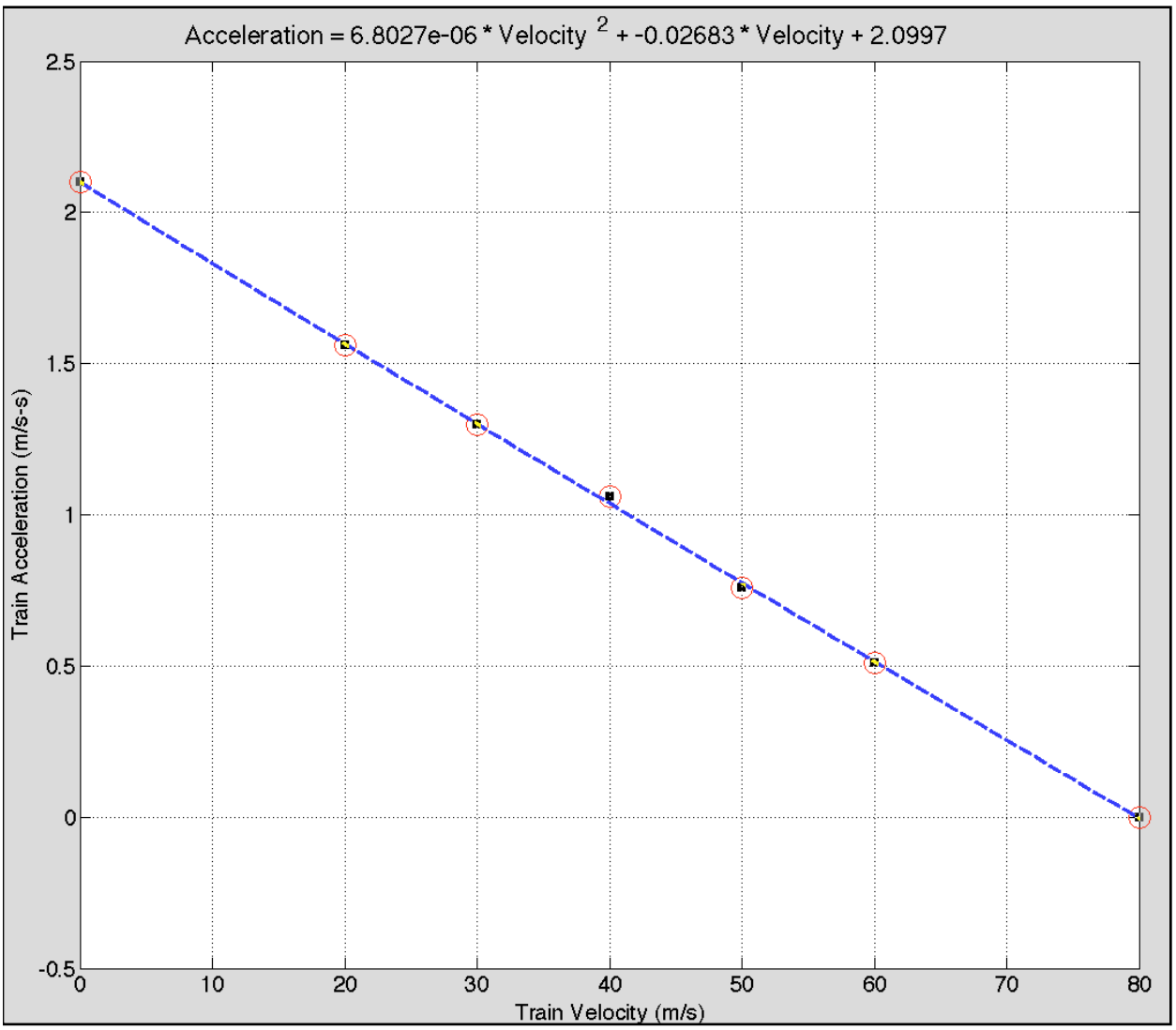

Figure 1. Plot of Train Velocity and Acceleration.

Task 2

Using the best fit polynomials estimated in Task 1, evaluate the goodness of fit of the polynomial by calculating the residuals or the local differences between the data points provided and the points of the acceleration function A(V) approximated by the fitted polynomial. Find the sum of the square errors of the fitted data and comment on the solution obtained. Is this a good fit? Tell me why.

```
% Task 2

% Find the residuals for each point. This requires that we find the values
% of acceleration from the polynomial fitted in Task 1

accelAtOriginalPoints = polyval(coefficients,velocity);

% Calculate the residuals = difference between original values and the
% fitted polynomial

residualValues = acceleration - accelAtOriginalPoints;
sumOfSquareErrors = sum(residualValues.^2);

disp(['Sum of the Square Errors = ', num2str(sumOfSquareErrors)])

% Make a plot of the residual values

figure
bar(velocity,residualValues)
xlabel('Train Velocity (m/s)','fontsize',20)
ylabel('Residual Values (m/s-s)','fontsize',20)
grid
```

The sum of the square errors is 8.068e-4. This value indicates that the fitted polynomial is a goos fit. The residuals are plotted in Figure 2. The largest discrepancy (difference between the original data and the polynomial fit) is observed at a value of velocity of 40 m/s. The difference is nevertheless small at less than 0.025 m/s-s indicating a good fit.

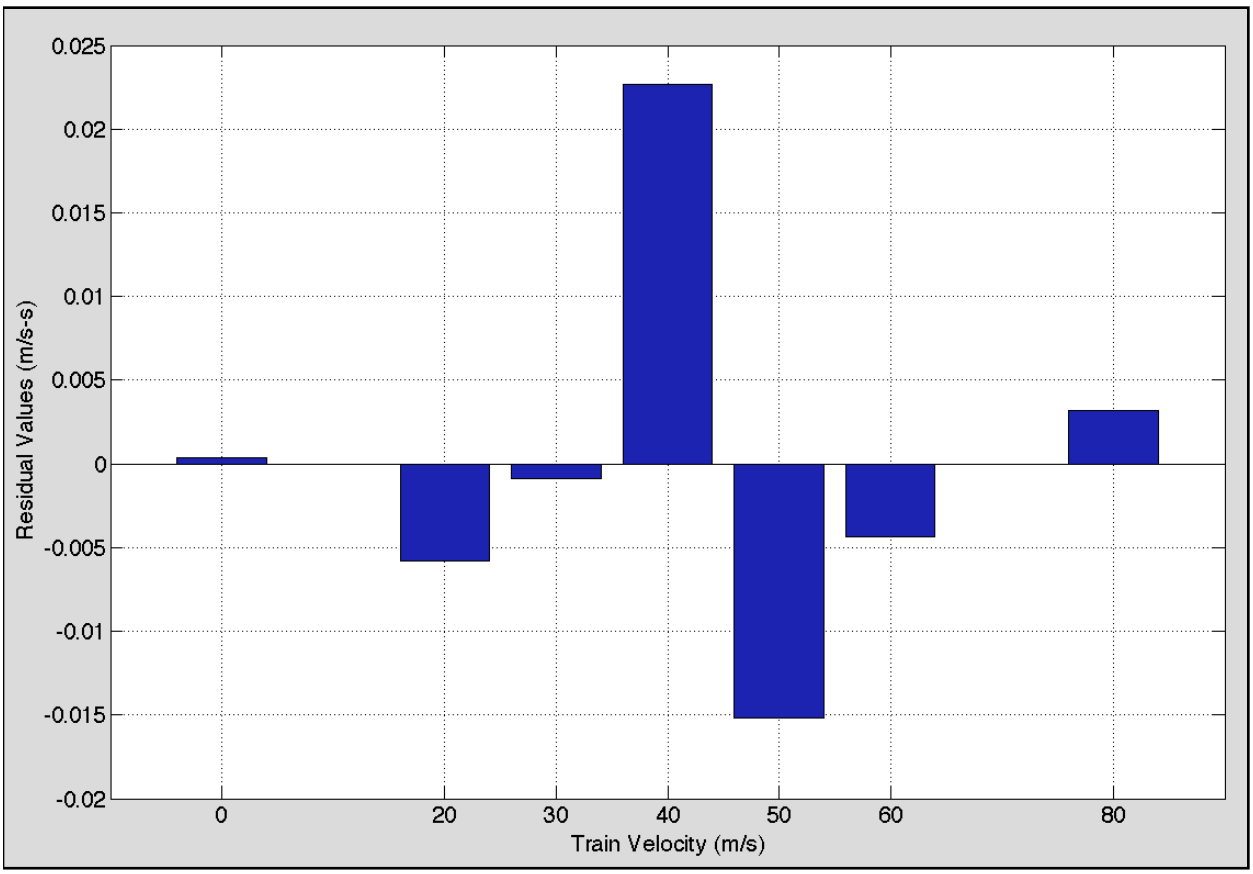

Figure 2. Residual Values of Train Acceleration.

Problem 4

A civil engineer sends you a "comma separated" data file containing aircraft characteristics used in airport design. The first few lines of the file look as follows:

Adam Aircraft Industries,A500,7000 ,98,44.00,9.58

Aero Spacelines,B-377,170000 ,123,156.25,48.50

Aeronca,11AC Chief,1250 ,64,36.00,6.00

Aeronca,15AC Sedan,2050 ,67,37.50,10.33

Aerospatiale,ATR-42-320,36817 ,124,80.58,24.92

The six columns of each record are explained below. The file name is "aircraft_characteristics_noheader_csv.csv".

Column 1 = aircraft name

Column 2 = manufacturer name

Column 3 = aircraft takeoff weight (lb)

Column 4 = aircraft approach speed (nautical miles per hour)

Column 5 = aircraft wingspan (feet)

Column 6 = aircraft tail height (feet)

Task 1

Create a Matlab script to read the complete data in this file in 6 column format. Use the Matlab "textscan" command with a comma delimiter. This information will be used in Task 3.

```
% Programmer: T. Trani (March 23, 2012)

clear
clc
% Read the data. The format of the data read is six columns seperated by cor

% Adam Aircraft Industries,A500,7000 ,98,44.00,9.58
% Aero Spacelines,B-377,170000 ,123,156.25,48.50
% Aeronca,11AC Chief,1250 ,64,36.00,6.00
% Aeronca,15AC Sedan,2050 ,67,37.50,10.33
% Aerospatiale,ATR-42-320,36817 ,124,80.58,24.92

fid = fopen('aircraft_characteristics_noheader_csv.csv');          % open the
readData = textscan(fid, '%s %s %f %f %f %f', 'delimiter', ',');   % read six c
fclose(fid)                                                        % closes th

% Define new variables before doing the analysis

manufacturer = readData{1};
name = readData{2};
takeoffWeight = readData{3};
approachSpeed = readData{4};
wingspan = readData{5};
tailHeight = readData{6};
```

Task 2
Create a Matlab function to assign a "wake" class (i.e., a string variable) and the airplane design group class (another string variable) to an aircraft according to their takeoff mass, wingspan and tail heights. According to the Federal Aviation Administration (FAA) the following rules are used by Air Traffic in assigning wake classes:

Small if takeoff weight < 41,000 lb
Large if takeoff weight >=41,000 lb and <= than 255,000 lb
Heavy if takeoff weight > 255,000 lb
Superheavy if takeoff weight > 1,000,000 lb

Similarly, the FAA groups aircraft according to their wingspan and tail height using the following letter designation (called aircraft design groups as shown in column 1 of the table):

Table 1-1. Airplane Design Groups (ADG)

Group #	Tail Height (ft)	Wingspan (ft)
I	<20	<49
II	20 - <30	49 - <79
III	30 - <45	79 - <118
IV	45 - <60	118 - <171
V	60 - <66	171 - <214
VI	66 - <80	214 - <262

The function should take as parameters the aircraft takeoff weight, the aircraft wingspan and the tail height and return two strings: a) wake class and b) the FAA airplane design group.

Solution:

The function findWakeClassV2 is shown in Figures 3 and 4. The function is shown in two figures because it does not fit the page completely. Figure 3 shows the first part of the Function findWakeClassV2. This section finds the wake class using the value of aircraft mass as single attribute. Figure 4 shows the second part of the Function findWakeClassV2. This section finds the aircraft design group (ADG) using the value of aircraft wingspan and aircraft tail height as attributes.

```
% Function to calculate the aircraft wake class and the aircraft design
% group (ADG).
% T. Trani (April 10, 2012)

% Inputs:

% 1) wingspan (feet)
% 2) mass (lb)
% 3 tail height (feet)

% Outputs:

% 1) wakeClass (i.e., small, large, heavy, superheavy)
% 2) Aircraft Design Group (ADG)  such as I through VI

function [wakeClass, ADG] = findWakeClassV2(mass, wingspan,tailheight)

if mass <  41000
   wakeClass = 'Small';
elseif mass >= 41000 && mass < 255000
   wakeClass = 'Large';
elseif mass >= 255000 && mass < 1e6
   wakeClass = 'Heavy';
else
   wakeClass = 'Superheavy';
end
```

Figure 3. First part of the Function findWakeClassV2. This section finds the wake class using the value of aircraft mass as single attribute.

```
% Now identify the aircraft design group (ADG)
% I use a temporary numerical value to assign an aircraft design group (intead of
% roman numerals). This value is called ADGw.
if wingspan <  49
   ADGw = 1;
elseif wingspan >= 49 && wingspan < 79
   ADGw = 2;
elseif wingspan >= 79 && wingspan < 118
   ADGw = 3;
elseif wingspan >= 118 && wingspan < 171
   ADGw = 4;
elseif wingspan >= 171 && wingspan < 214
   ADGw = 5;
else
   ADGw = 6;
end

% Now identify the aircraft design group (ADG) based on tail height
% I use a numerical value to assign an aircraft design group (intead of
% roman numerals). This value is called ADGt.

if tailheight <  20
   ADGt = 1;
elseif tailheight >= 20 && tailheight < 30
   ADGt = 2;
elseif tailheight >= 30 && tailheight < 45
   ADGt = 3;
elseif tailheight >= 45 && tailheight < 60
   ADGt = 4;
elseif tailheight >= 60 && tailheight < 66
   ADGt = 5;
else
   ADGt = 6;
end

% Since there are two criteria for ADG we select the most demanding of the two
% Let a table of ADG design groups with Roman Numeral strings be constructed as follows:
ADGtable = {'I'; 'II'; 'III'; 'IV' ; 'V'; 'VI'};
ADGnumerical = max(ADGt, ADGw);        % selects the maximum number of the two ADG values

ADG = ADGtable{ADGnumerical};          % assigns the correct Roman numeral string value to ADG
```

Figure 4. Second part of the Function findWakeClassV2. This section finds the aircraft design group (ADG) using the value of aircraft wingspan and aircraft tail height as attributes.

Testing the findWakeClassV2 function from the command line.

[wakeClass, ADG] = findWakeClassV2(61000, 100,20) produces

wakeClass = Large and **ADG = III**

[wakeClass, ADG] = findWakeClassV2(610000, 200,80) produces

wakeClass = Heavy and **ADG = VI**

Task 3
Modify the Matlab script created in Task 1 and using the function created in Task 2, assign the wake classes and airplane design groups to the complete list of aircraft read in Task 1.

```
% Task 3 - assign wake class and ADG group to each aircraft read form the
% data file. This requires a for-loop that calls the function
% findWakeClassV2 318 times - one time for each aircraft in the data set.

noAircraft = length(takeoffWeight);          % detect the number of aircraft in data set

for i=1:noAircraft

   % call the findWakeClassV2 function here to obtain two string
   % variables: wakeClass and ADG. These variables are reused every time
   % the function executes.

   [wakeClass , ADG] = findWakeClassV2(takeoffWeight(i), wingspan(i), tailHeight(i));

   % Save the values of the two string variables into two celll arrays with 318 entries.
   % Remember, these are strings and as such need to be saved in a cell
   % array.

   aircraftWake{i} = wakeClass;
   aircraftDesignGroup{i} = ADG;

end
```

Figure 5. Script to find and assign wake class and ADG to each element of the aircraft data set.

Task 4
Modify the Matlab script created in Task 3 and using the string comparison function in Matlab extract all aircraft that are class “Large” and that belong to airplane design group III. Plot their aircraft wingspan and tail height.

```
% Task 4 - Find all the large aircraft using the string comparison function

% The line 56 in the program finds indices (0 or 1) for large aircraft in the data set using the string
% comparison function in Matlab (as required). index = 1 if a match is
% found, 0 otherwise.

indicesForLargeAircaft = strcmp(aircraftWake,'Large');

largeAircraftWingspan = wingspan(indicesForLargeAircaft);     % saves wingspan values for large aircraft
largeAircraftTailheight = tailHeight(indicesForLargeAircaft);     % saves tail height values for large aircraft

% Plot wingspan vs. tail height

figure
plot(largeAircraftWingspan,largeAircraftTailheight,'or')
xlabel('Aircraft Wingspan (ft)','fontsize',20)
ylabel('Tail Height (ft)','fontsize',20)
grid
```

Figure 6. Script to match the string 'Large' to each element of vector aircraft wake (found in Figure 3). This extracts the aircraft wingspan and tail height of "large" aircraft in the data set. A total of 109 Large aircraft were found.

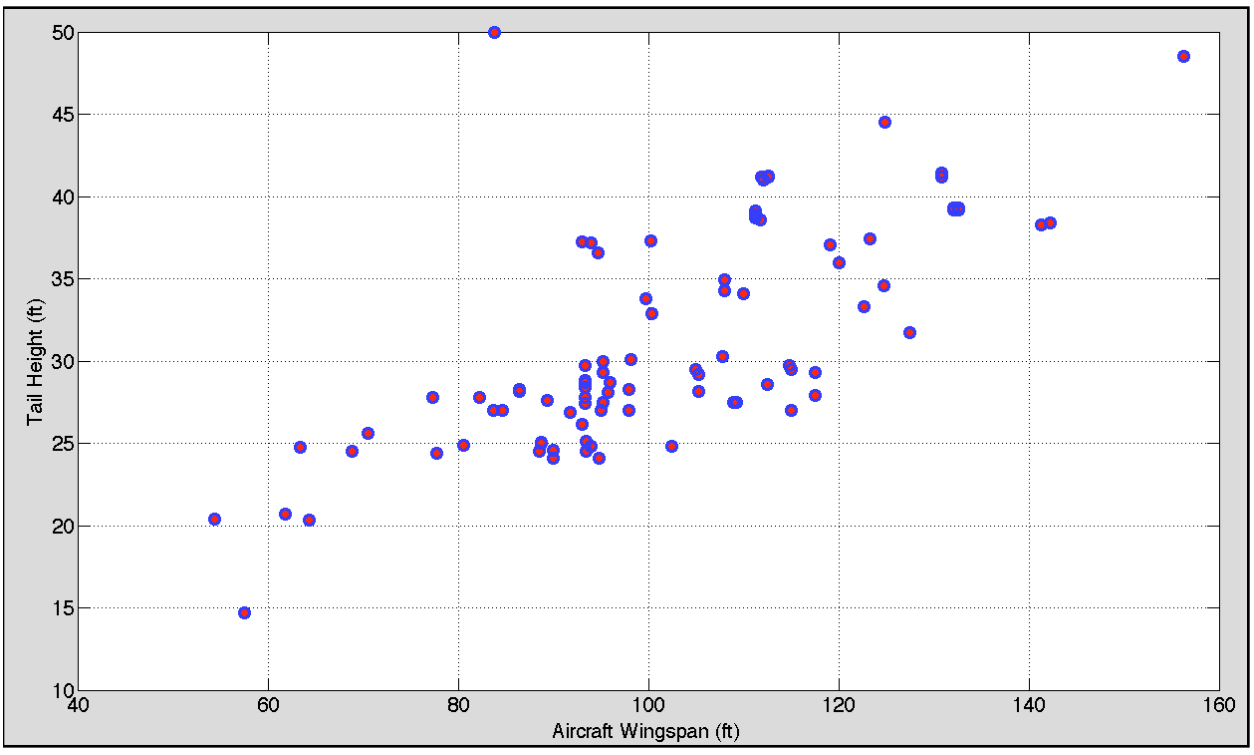

Figure 7. Plot of aircraft wingspan and tail height.