Programming in LabVIEW MathScript-A Data Analysis Example Using For Loops*

Anthony Antonacci
Darryl Morrell

Based on *Programming in MATLAB-A Data Analysis Example Using For Loops*† by Darryl Morrell

Abstract

This is an example of using LabVIEW MathScript to solve a simple engineering data analysis problem in which velocity and acceleration are computed from altitude data from a home-built rocket test launch.

NOTE: This example requires an understanding of the relationships between position, velocity, and acceleration of an object moving in a straight line. The Wikipedia article *Motion Graphs and Derivatives*[1] has a clear explanation of these relationships, as well as a discussion of average and instantaneous velocity and acceleration and the role derivatives play in these relationships. Also, in this example, we will approximate derivatives with forward, backward, and central differences; *Lecture 4.1* by Dr. Dmitry Pelinovsky at McMaster University [2] contains useful information about this approximation. We will also approximate integrals using the trapezoidal rule; The Wikipedia article *Trapezium rule*[3] has an explanation of the trapezoidal rule.

1 Trajectory Analysis of an Experimental Homebuilt Rocket

On his web page *Richard Nakka's Experimental Rocketry Web Site: Launch Report - Frostfire Two Rocket*[4] , Richard Nakka provides a very detailed narrative of the test firing of his Frostfire Two homebuilt rocket and subsequent data analysis. (His site[5] provides many detailed accounts of tests of rockets and rocket motors.

*Version 1.1: Aug 2, 2006 1:19 pm -0500
†http://cnx.org/content/m13277/1.5/
‡http://creativecommons.org/licenses/by/2.0/
[1]http://en.wikipedia.org/wiki/Motion_graphs_and_derivatives
[2]http://dmpeli.math.mcmaster.ca/LabVIEW MathScript/Math1J03/LectureNotes/Lecture4_1.htm
[3]http://en.wikipedia.org/wiki/Trapezoidal_rule
[4]http://www.nakka-rocketry.net/ff-2.html
[5]http://www.nakka-rocketry.net/

Some rocket launches were not as successful as the Frostfire Two launch; his site provides very interesting post-flight analysis of all launches.)

2 Computation of Velocity and Acceleration from Altitude Data

In this section, we will use LABVIEW MATHSCRIPT to analyze the altitude data extracted from the plot "Altitude and Acceleration Data from R-DAS" on Richard Nakk's web page. This data is in the file Altitude.csv[6] . We will use this data to estimate velocity and acceleration of the Frostfire Two rocket during its flight.

Exercise 1

Get the data

Download the altitude data set in the file Altitude.csv[7] (right click on the file's link) onto your computer. Then be sure to save the file to your MATHSCRIPT working directory. This can be determined by navigating to the File/MathScript Perferences tab. The file is formatted as two columns: the first column is time in seconds, and the second column is altitude in feet. Load the data into LABVIEW MATHSCRIPT and plot the altitude as a function of time.

The following sequence of LABVIEW MATHSCRIPT commands will load the data, create a vector `t` of time values, create a vector `s` of altitude values, and plot the altitude as a function of time.

```
    Altitude = csvread('Altitude.csv');
t =Altitude(:,1);
s =Altitude(:,2);
plot(t,s)
```

The plot should be similar to that in Figure 1.

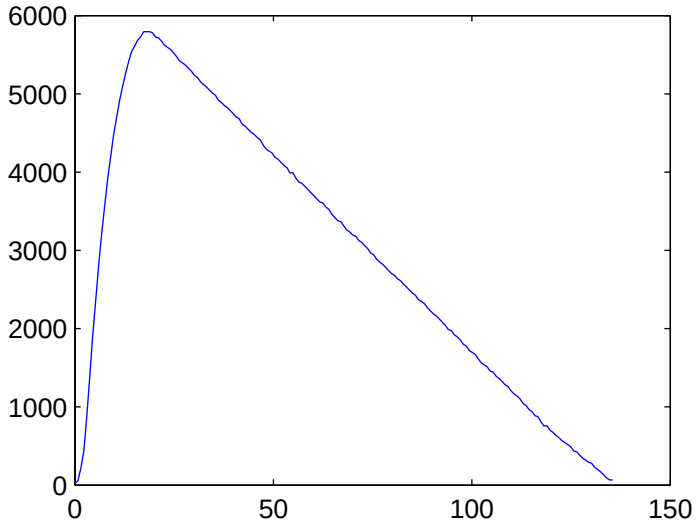

Figure 1: Plot of altitude versus time.

[6] http://cnx.org/content/m13715/latest/Altitude.csv
[7] http://cnx.org/content/m13715/latest/Altitude.csv

Exercise 2 *(Solution on p. 5.)*

Forward Differences

Write a LABVIEW MATHSCRIPT script that uses a for loop to compute velocity and acceleration from the altitude data using forward differences. Your script should also plot the computed velocity and acceleration as function of time.

Exercise 3 *(Solution on p. 5.)*

Backward Differences

Modify your script from Problem (Forward Differences) to compute velocity and acceleration using backward differences. Remember to save your modified script with a different name than your script from Problem (Forward Differences).

Exercise 4 *(Solution on p. 6.)*

Central Differences

Modify your script from Problem (Forward Differences) to compute velocity and acceleration using central differences. Remember to save your modified script with a different name than your script from Problem (Forward Differences) and Problem (Backward Differences).

Compare the velocity and acceleration values computed by the different approximations. What can you say about their accuracy?

Exercise 5 *(Solution on p. 6.)*

Can it be done without loops?

Modify your script from Problem (Forward Differences) to compute velocity and acceleration without using a for loop.

3 Computation of Velocity and Altitude from Acceleration Data

In this section, we will use LABVIEW MATHSCRIPT to analyze the acceleration data extracted from the plot "Altitude and Acceleration Data from R-DAS" on Richard Nakk's web page. Download the acceleration data set in the file Acceleration.csv[8] (right click on the file's link) onto your computer. The first column is time in seconds, and the second column is acceleration in g's. The following commands load the data into LABVIEW MATHSCRIPT and plot the acceleration as a function of time.

```
    Acceleration = csvread('Acceleration.csv');
t = Acceleration(:,1);
a = Acceleration(:,2);
plot(t,a)
```

The plot should be similar to that in Figure 2.

[8] http://cnx.org/content/m13715/latest/Acceleration.csv

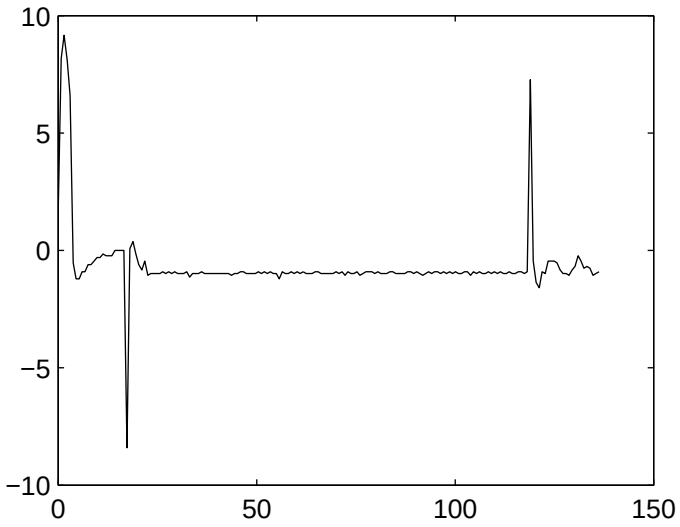

Figure 2: Plot of acceleration versus time.

Exercise 6 *(Solution on p. 7.)*

Trapezoidal Rule

Write a LABVIEW MATHSCRIPT script that uses a for loop to compute velocity and altitude from the acceleration data using the trapezoidal rule. Your script should also plot the computed velocity and altitude as function of time.

Exercise 7 *(Solution on p. 9.)*

Can it be done without loops?

Modify your script from Problem (Trapezoidal Rule) to compute velocity and altitude without using a for loop.

Solutions to Exercises in this Module

Solution to Exercise (p. 2)

This solution is by Scott Jenne; it computes and plots the velocity:

```
    Altitude = csvread('Altitude.csv');
t=Altitude(:,1);
s=Altitude(:,2);
for n=1:180;
    v=((s(n+1))-s(n))/((t(n+1))-t(n));
    hold on
    plot(t(n),v,'o')
end
```

The plot produced by this code is shown in Figure 3.

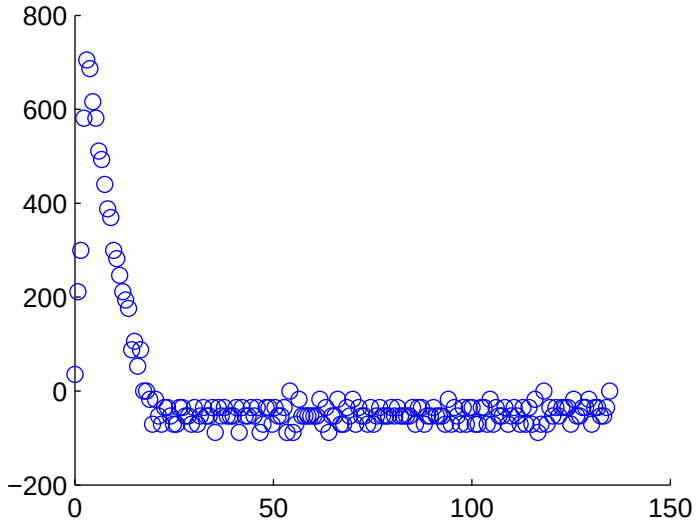

Figure 3: Plot of velocity computed with the forward difference method versus time.

Solution to Exercise (p. 3)

This solution by Bryson Hinton:

```
    Altitude = csvread('Altitude.csv');
t=Altitude(:,1);
s=Altitude(:,2);
for x=2:181
    v(x)=(s(x)-s(x-1))/(t(x)-t(x-1));
    plot(t(x),v(x),'b.')
    hold on
end
```

The plot produced by this code is shown in Figure 4.

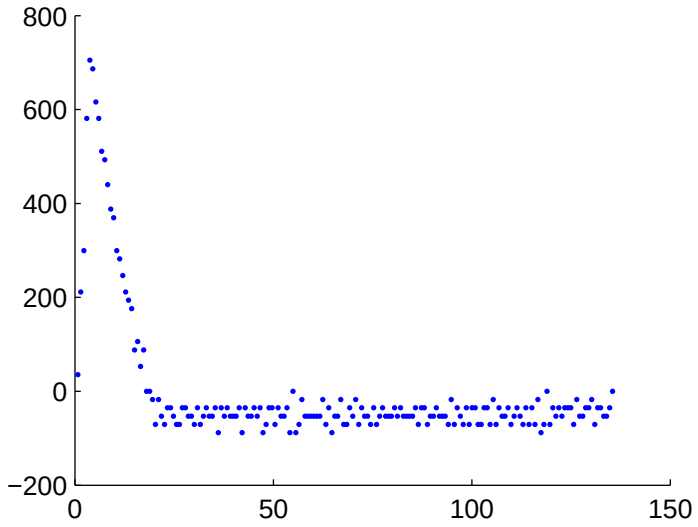

Figure 4: Plot of velocity computed with the backward difference method versus time.

Solution to Exercise (p. 3)
This code computes the velocity using the central difference formula.

```
    clear all
Altitude = csvread('Altitude.csv');
t=Altitude(:,1);
s=Altitude(:,2);
for n=2:180
    v(n-1)=(s(n+1)-s(n-1))/(t(n+1)-t(n-1));
end
plot(t(2:180),v)
```

The plot produced by this code is shown in Figure 5.

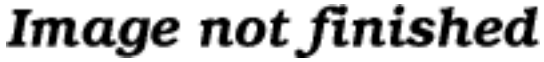

Figure 5: Plot of velocity computed with the central difference method versus time.

Solution to Exercise (p. 3)
This code uses LABVIEW MATHSCRIPT's `diff` function to compute the difference between adjacient elements of `t` and `s`, and the `./` function to divide each element of the altitude differences with the corresponding element of the time differences:

```
    clear all
Altitude = csvread('Altitude.csv');
t=Altitude(:,1);
s=Altitude(:,2);
v=diff(s)./diff(t);
plot(t(1:180),v)
```

The plot produced by this code is shown in Figure 6.

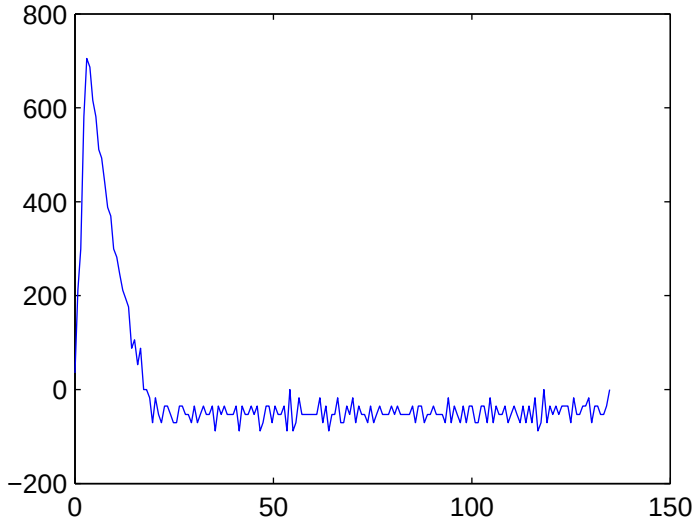

Figure 6: Plot of velocity computed with the forward difference method versus time. The values in this plot are the same as in Figure 3.

Solution to Exercise (p. 4)
This solution is by Jonathan Selby:

```
    Acceleration = csvread('Acceleration.csv');
t=Acceleration (:,1);
a=Acceleration (:,2);
v(1)=0;
for n=1:181
    v(n+1)=(t(n+1)-t(n))*(a(n+1)+a(n))/2+v(n);
end
plot(t,v)
```

This code creates the plot in Figure 7.

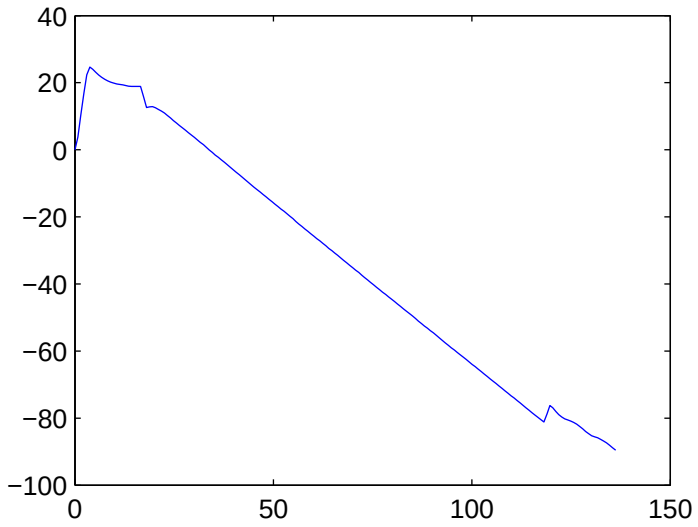

Figure 7: Plot of velocity versus time. The velocity is computed by numerically integrating the measured acceleration.

This code can be easily extended to also compute altitude while it is computing velocity:

```
    Acceleration = csvread('Acceleration.csv');
t=Acceleration (:,1);
a=Acceleration (:,2);
v(1)=0; % Initial velocity
s(1)=0; % Initial altitude
for n=1:181
    v(n+1)=(t(n+1)-t(n))*(a(n+1)+a(n))/2+v(n);
    s(n+1)=(t(n+1)-t(n))*(v(n+1)+v(n))/2+s(n);
end
plot(t,s)
```

This code creates the plot in Figure 8.

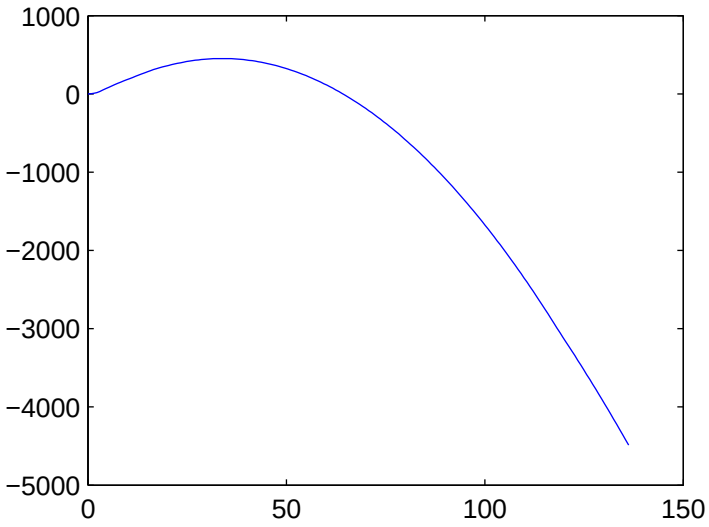

Figure 8: Plot of altitude versus time.

Solution to Exercise (p. 4)
This solution by Nicholas Gruman uses the LABVIEW MATHSCRIPT `trapz` function to compute velocity with the trapezoidal rule with respect to time at all given points in time:

```
    Acceleration = csvread('Acceleration.csv');
t=Acceleration(:,1);
A=Acceleration(:,2);
c = length(t);
for i = 1:c
     v(i)=trapz(t(1:i),A(1:i));
end
```

Altitude could also be computed by adding the following line in the same For loop as the previous code:

```
    s(i)=trapz(t(1:i),v(1:i));
```