Armstrong State University
Engineering Studies
MATLAB Marina – Integration Primer

Prerequisites

The Integration Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, arrays, logic expressions, conditional structures, iteration, functions, debugging, characters and strings, cell arrays, structures, file input and output, 2D plotting, 3D plotting, interpolation, curve fitting, and numerical differentiation. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, MATLAB Marina Arrays module, MATLAB Marina Logic Expressions module, MATLAB Marina Conditional Structures module, MATLAB Marina Iteration module, MATLAB Marina Functions module, MATLAB Marina debugging module, MATLAB Marina Character and Strings module, MATLAB Marina Cell Arrays module, MATLAB Marina Structures module, MATLAB Marina File Input and Output module, MATLAB Marina Plotting module, MATLAB Marina Interpolation Module, MATLAB Marina Curve Fitting Module, and MATLAB Marina Differentiation Module.

Learning Objectives

1. Be able to use MATLAB to compute numerical integrals.
2. Understand the limitations of numerical integration.

Terms

numerical integration, complete (definite) integral, continuous integral, noisy data

MATLAB Functions, Keywords, and Operators

sum, trapz, cumsum, cumtrapz, quad, quad2, integral, integral2
Note: quad, quad2, integral, integral2 are not used in ENGR 1371 but are used in ENGR 2010 and are not covered in this primer.

Numerical Integration

The definite integral of a function $f(x)$ can be interpreted as an area $R = \int_a^b f(x)\,dx$ and the indefinite integral of a function can be interpreted as the anti-derivative of a function $f(x) = \int \frac{df(x)}{dx}\,dx$. Similar to numerical differentiation, numerical integration is typically used when a formula for the function to integrate is not available or the integral would be difficult to determine analytically. Numerical integration can also be used to compute estimates of integrals that may be solved analytically later. Numerical integration can yield accurate results if the function is piecewise continuous, relatively smooth, and the increment between function points is relatively small. Numerical integration typically yields better results than numerical differentiation as it is less sensitive to inaccuracies in the data (noisy data).

The Trapezoidal rule and Simpson's rule are two common methods of performing numerical integration for definite integrals. For the Trapezoidal rule, the area between the function and the x-axis is approximated by a trapezoid.

$$\int_a^b f(x)\,dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

For numerical integration using the composite Trapezoidal rule, the interval of integration is split into N small uniform subintervals, each approximated by a trapezoid, and the area of all the trapezoids is summed.

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2N}\left(f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{N-1})+f(x_N)\right)$$

Where the x_k are the end points of the trapezoids

$$x_k = a + k\frac{b-a}{N},\ k = 0,1,\cdots,N$$

For numerical integration using Simpson's rule, the interval is approximated by a quadratic function, and the integral is approximated by the area between the quadratic functions and the x-axis.

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

Breaking the interval up into 2N subintervals and approximating each interval by a quadratic function, and summing the areas between the quadratic functions and the x-axis, the integral is approximated by the composite Simpson's rule

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)\cdots+2f(x_{2N-2})+4f(x_{2N-1})+f(x_{2N})\right)$$

Where the x_k are the end points of the trapezoids

$$x_k = a + k\frac{b-a}{2N},\ k = 0,1,\cdots,2N$$

and

$$h = \frac{b-a}{2N}$$

Cumulative numerical integration is used when we want to determine an approximation of the anti-derivative of a function.

$$\int_{-\infty}^{t} f(\tau)\,d\tau$$

Numerical Integration using MATLAB

The MATLAB function `trapz` is used to approximate a definite integral of a function using the Trapezoidal rule. The `trapz` function takes two arguments x and y and returns the integral of y

with respect to x. The vectors x and y must be the same length and the interval for the approximate definite integral is specified via the x vector values.

The MATLAB functions cum`trapz` and `cumsum` are used to approximate the anti-derivative of a function (cumulative numerical integral). The cum`trapz` function takes two arguments x and y and returns the cumulative integral of y with respect to x using the trapezoidal method. The vectors x and y must be the same length, the vector x should have uniform spacing, and the result is a vector the same size as x and y. If the spacing in x is one (distance between x values is one), then `cumtrapz` can be used with a single argument y.

The MATLAB program of Figure 1a computes the approximate definite integral of $\int_{-5}^{5}\left(x^2+2x+1\right)dx$ and cumulative integral (approximate anti-derivative) of $\int_{-5}^{x}\left(u^2+2u+1\right)du$.
Figure 1b shows the function and its cumulative integral.

```
% function f(x) = x^2 + 2x + 1
coef = [1, 2, 1];
x = -5 : 0.1 : 5;
f = polyval(coef, x);

% approximate definite integral of f(x) from -5 to 5
defintf = trapz(x,f);
% approximate anti-derivative of f(x)
cumintf = cumtrapz(x,f);

figure(1)
plot(x,f,'b-',x,cumintf,'g-')
xlabel('x'), legend('f(x)','approximate anti-derivative of f(x)')
```

Figure 1a, MATLAB Program Performing Numerical Integration

If the spacing in x is not uniform, then `cumtrapz(y)` can be used to compute the cumulative integral with unit spacing. This result must then be element by element multiplied with the spacings. The `cumsum` function can be used similar to `cumtrapz` to perform cumulative numerical integration except it only takes one argument and must either be multiplied by the spacing increment or element by element multiplied with the spacings for non-uniform spacing.

Remember that the MATLAB function `trapz` is for computing approximate definite integrals and the result will be a scalar and that `cumtrapz` is for computing cumulative (indefinite) integrals and the result will be a vector.

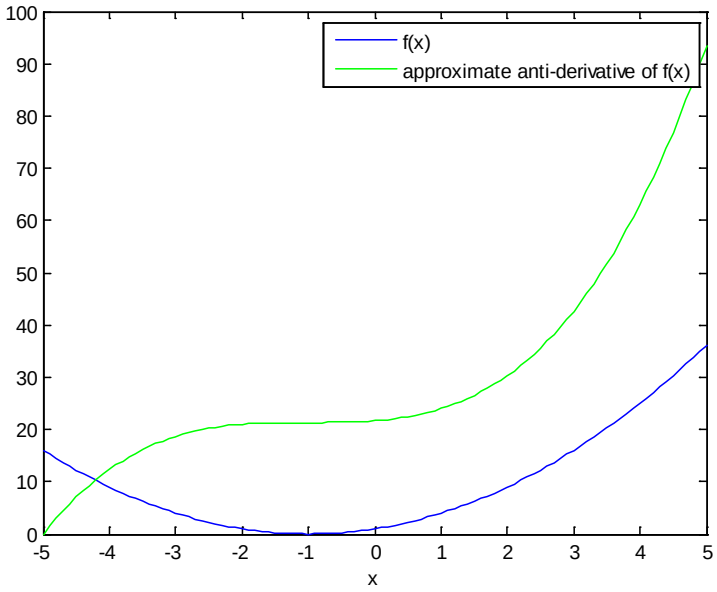

Figure 1b, Resulting Plot of Cumulative Integration Example

Numerical Integration and Differentiation of Noisy Data

One must be careful when performing numerical operations on noisy data (data with uncertainties or data with measurement error). The noise or error associated with measured or uncertain data is often small in magnitude but rapidly changing.

Integration tends to reduce the effect of error or noise whereas differentiation tends to magnify the effect of error or noise. Integration is similar to averaging and tends to reduce the effects of things that are small but vary rapidly. Differentiation which gives the rate of change of something, magnifies things that vary rapidly even if the magnitude of the variation is small. The derivative of the error or noise in data may be larger in magnitude than the derivative of the underlying function of the data.

Figures 2a shows a MATLAB program that generates a noisy signal and determines the approximate integral and approximate derivative. The underlying function is a unit amplitude cosine of 250Hz. The additive noise is a cosine of 60Hz with amplitude uniformly distributed over the range -0.05 to 0.05.

Figures 2b -2d show plots of the noisy data, the approximate integral of the noisy data, and the approximate derivative of the noisy data. Notice in the plot of Figure 2b, that the noise in the data is barely discernible except at a few of the peaks. The signal looks like a unit amplitude cosine of 250Hz (period 4msec).

```
% noisy data, 250Hz cosine with additive noise
% modeled as a 60Hz cosine with varying amplitude
t = 0:1/250/20:0.02;
% unit amplitude cosine of 250Hz
g = 1.0*cos(2*pi*250*t);
% noise, random amplitude 60Hz cosine
noise = (-0.05 + (0.05 - -
0.05)*rand(1,length(t))).*cos(120*pi*t);
% data plus noise
gNoisy = g + noise;

% approximate integral and derivative of the noisy data
intgNoisy = cumtrapz(t,gNoisy);
dgNoisy = diff(gNoisy)./diff(t);
```

Figure 2a, Numerical Integral and Approximate Derivative of Noisy Data

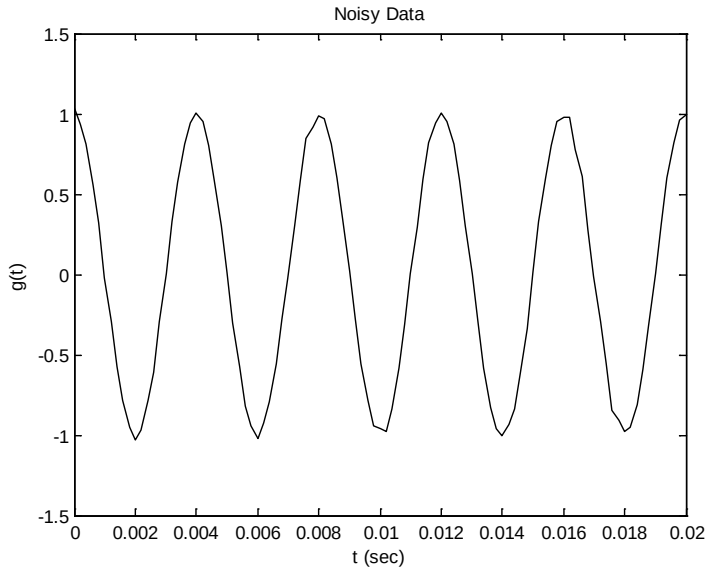

Figure 2b, Plot of Noisy Data

The integral of the noisy data, shown in Figure 2c, is a sin of approximately 250Hz. The amplitude is much smaller than one might expect, but recall that $\int \cos(\omega t)dt = \frac{1}{\omega}\sin(\omega t) + c$, so the amplitude the sine wave should be scaled by $\frac{1}{\omega}$ times the amplitude of the noisy signal.

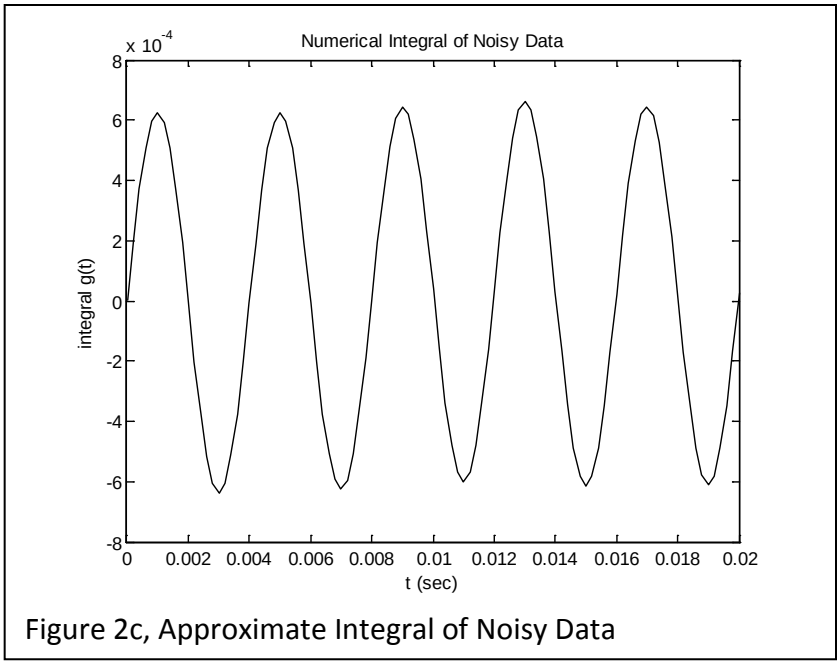

Figure 2c, Approximate Integral of Noisy Data

The derivative of the noisy data, Figure 2d, should be a negated sin scaled by ω. The plot is similar to this but the initial value and frequency are off. There are five periods of the original signal in 20msec but only around 4.5 periods of the signal's derivative. Notice that in several areas the derivative of noise is easily apparent (all the negative peaks, second through fifth positive peaks, down slope of second peak, etc.).

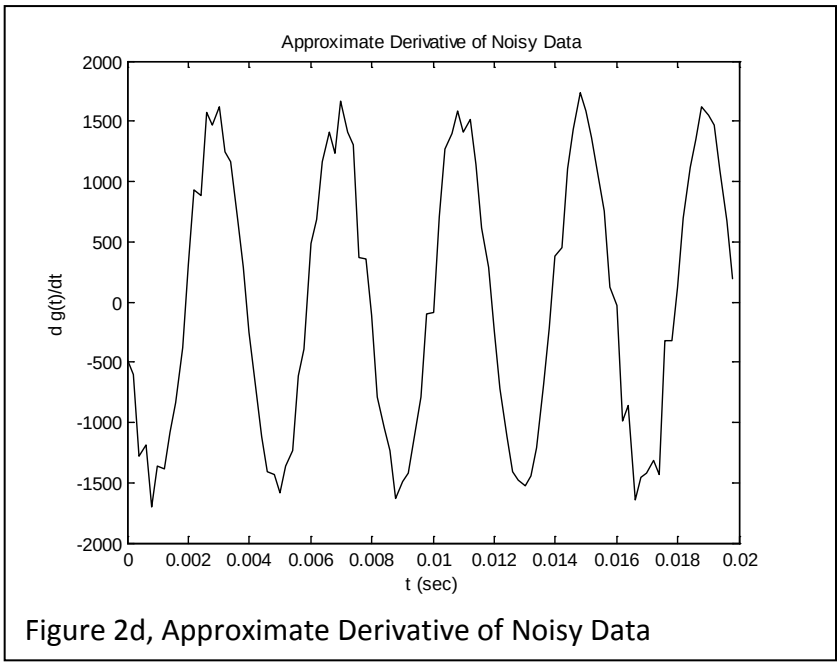

Figure 2d, Approximate Derivative of Noisy Data

In this example the magnitude of the noise was 5% or less of the original signal and was multiplied by a cosine of 60Hz which for most points further reduced the magnitude of the noise. The plots of Figures 3a and 3b shows the effect of purely additive noise. Changing the state in the program Figure 2a the computes the noise to

```
noise = (-0.05 + (0.05 - -0.05)*rand(1,length(t)));
```

The effect of the noise is barely noticeable on the approximate integral but more pronounced on the approximate derivative.

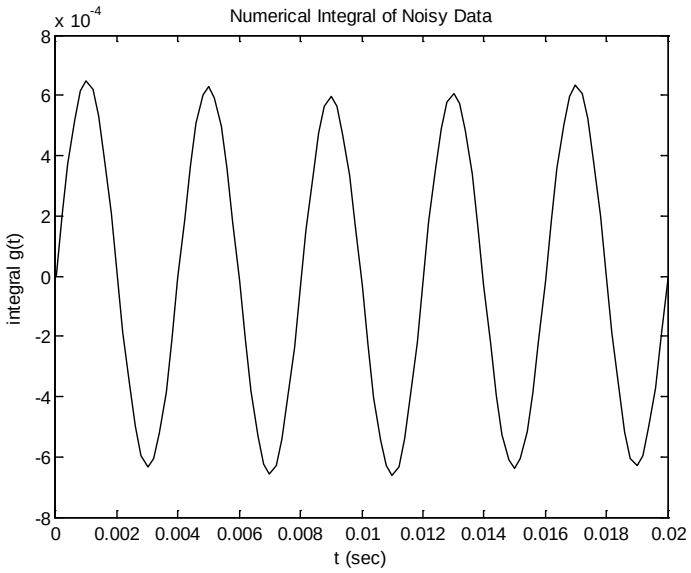

Figure 3a, Approximate Integral of Noisy Data

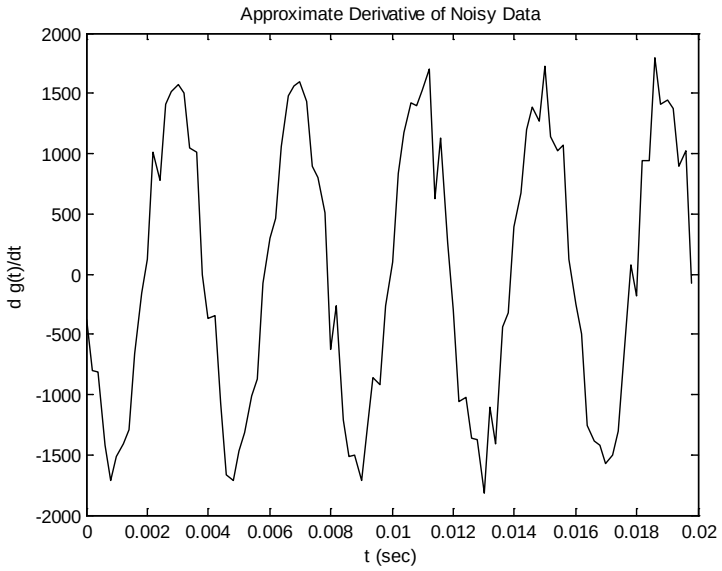

Figure 3b, Approximate Derivative of Noisy Data

Last modified Thursday, November 13, 2014