GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2025 Sring 2001
Lab #10: Numerical Evaluation of Fourier Series

Date: 27-29 March 2001

Lab Quiz #2 will be held at the beginning of this Lab.

This is *the official* Lab #10 description.

The Warm-up section of each lab must be completed in Lab and the steps marked *Instructor Verification* must also be signed off **during your scheduled lab time**.

Lab Report: Please turn in Section 4 with explanations as this week's informal lab report.

The report will **due during the week of 2-6 April at the start of your lab**.

---

1 Introduction & Objective

The goal of the laboratory project is to show how Fourier Series analysis can be done numerically without analytically evaluating integrals, when the input is a periodic signal. Since we will be doing Fourier Series for continuous-time signals, the formulas will be integrals, but we will use MATLAB's numerical integration capability to evaluate the integrals without doing the integrals by hand. (Another approach would be to use a symbolic algebra package such as *Mathematica* or Maple to derive formulas for the Fourier Series coefficients.)

In the next lab, we will use Fourier Series to analyze a frequency synthesis design problem in the frequency domain.

2 Background: Fourier Series Analysis and Synthesis

Recall the *analysis* integral and *synthesis* summation for the Fourier Series expansion of a periodic signal $x(t) = x(t + T_0)$. The Fourier synthesis equation for a periodic signal $x(t) = x(t + T_0)$ is

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}, \tag{1}$$

where $\omega_0 = 2\pi/T_0$ is the *fundamental* frequency. To determine the Fourier series coefficients from a signal, we must evaluate the *analysis* integral for every integer value of k:

$$a_k = \frac{1}{T_0} \int_0^{T_0} x(t) e^{-jk\omega_0 t} dt \tag{2}$$

where $T_0 = 2\pi/\omega_0$ is the *fundamental* period. If necessary, we can evaluate the analysis integral over any period; in (2) the choice $[0, T_0]$ was a convenient one, but integrating over the interval $[-\frac{1}{2}T_0, \frac{1}{2}T_0]$ would also give exactly the same answer.

3 Warmup

In this lab, we will use a feature of MATLAB that allows you to evaluate integrals by numerical approximation. MATLAB has several numerical integration functions, such as `quad()` and `quad8()` that are very powerful methods for evaluating integrals of functions specified by formulas. The formula has to be a MATLAB function, but the numerical integration function will adaptively figure out the best way to approximate the integral with a sum. We can use these functions to evaluate Fourier Series integrals and then we will plot the resulting Fourier Series coefficients $\{a_k\}$ to display the spectrum.

3.1 Numerical Integration

The basic idea of numerical integration is to approximate an integral with a sum, and the simplest way to do this is with the Riemann sum:

$$\int_a^b f(t)dt \approx \sum_{n=1}^{L} f(t_n)\left(\frac{b-a}{L}\right) \tag{3}$$

where the "sampling points" t_n are $t_n = a + (n - \frac{1}{2})\Delta$, and $\Delta = \frac{b-a}{L}$. The parameter Δ is actually the width of subintervals defined by breaking the interval $[a, b]$ into L equal sized subintervals. The sampling points are taken to be the *mid-points* of those subintervals.[1]

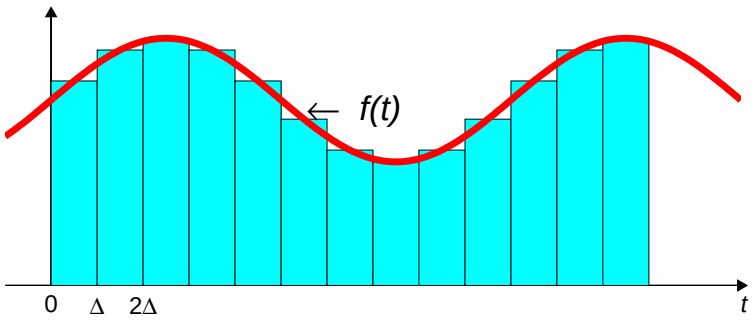

Figure 1: Approximating the area under $f(t)$ with the area of many narrow triangles.

The theory of Riemann integration (in calculus) tells us that taking the limit as $L \to \infty$ will make the sum converge to the integral in (3). Furthermore, if we think of the integral as calculating area, then the Riemann sum is approximating the area with the sum of areas of many rectangles whose width is Δ and whose height is $f(t_n)$. We could get a better approximation to the area if we used trapezoids instead of rectangles to compute the area under the curve $f(t)$. Carrying this idea to the next step, we can use polynomial approximations to $f(t)$ to get even more accurate area approximations. This is what the MATLAB functions `quad` and `quad8` will do. Consult `help quad` or `help quad8` for more information.

3.2 Integrate a Simple Function

Suppose that we want to calculate the value of the definite integral[2]

$$\int_0^5 \cos(\pi t)dt = ?$$

[1] We could define t_n to be at one of the endpoints of the interval, and the same results would be obtained for approximating the integral with a sum.

[2] This easy example was chosen so that you can check the result by hand.

In order to use `quad8('f',a,b)` we must pass three arguments: the function definition (of the integrand), and also the lower and upper limits of integration. The limits of integration are scalar constants, so the tricky part is passing the function definition. There are two ways to do this: (1) write a separate MATLAB function to define the integrand, or (2) use the MATLAB function `inline()` to define the function via a string.

In MATLAB it seems reasonable that we might be able to write `'cos(pi*t)'` as a string and then try to pass the string as the first argument, but this won't work without the `inline()` "wrapper." On the other hand, it is possible to define the integrand as a function that can be evaluated. The primary way to do this is to write a separate M-file. For example, we might create the auxiliary function in an M-file called `fcos()` by writing the following one line function:

```
function out = fcos(t)
%FCOS   function define for integrating a cosine
%
out = cos(pi*t);
```

Once we have the `fcos()` function stored in the file `fcos.m`, we can perform the integration by using the **name** of the function in the call to `quad8()`:

```
quad8( 'fcos', 0, 5 )
```

You should run this example and verify that it gives the correct answer (because you can do the integral in your head). Notice that MATLAB gives an answer that is correct to the inherent numerical precision of double-precision floating point which is usually about 15 digits of accuracy.

(a) Modify the example above to do the integral of e^t from $t = 0$ to $t = 2$. This requires that you write a new auxiliary function M-file.

(b) Show that you can program "cases" into your function definition by doing the integral from $t = 0$ to $t = 4$ of the signal

$$f(t) = \begin{cases} 2 - t & \text{for } 0 \le t < 2 \\ t - 2 & \text{for } 2 \le t \le 4 \end{cases} \quad = \quad (2-t)[u(t) - u(t-2)] + (t-2)[u(t-2) - u(t-4)] \qquad (4)$$

This is easy enough that you can figure out the correct answer by hand to verify your work.[3]

Pay careful attention to the warning in the help for `quad8` that states:

```
Q = QUAD8('F',A,B) approximates the integral of F(X) from A to B...
'F' is a string containing the name of the function.  The function
must return a vector of output values if given a vector of input values.
```

In other words, you must vectorize the cases; perhaps by using the `find` function or vectorized logical tests. For example, you can define $f(t) = e^t[u(t) - u(t-1)]$ by writing

```
function out = myfunc(t)
out = exp(t).*(t>=0 & t<=1);
```

Note that the logical statement `(t>=0 & t<=1)` evaluates to 1 if the condition is true and it evaluates to 0 if the condition is false. Thus, multiplying by `(t>=0 & t<=1)` has the effect of "switching on" the exponential function over the desired range of values of `t`. You can use this technique to switch on and off various pieces of the function definition as required in (4)

Instructor Verification (separate page)

(c) A useful plotting function when you have a function definition programmed into an M-file is `fplot()`. Use `fplot()` to show[4] the function defined in (4) over the range $-1 \le t \le 5$.

[3] You might see warnings when `quad8` runs; these seem to be caused by the corner in $f(t)$ at $t = 2$.

[4] Type `help fplot` to learn the usage of `fplot`.

3.3 Inline Functions

We can eliminate the auxiliary M-file if we exploit MATLAB's capability to have "in-line" function definitions. This only works when the function is simple and can be expressed as a "one-liner." But that does apply to the cosine example above, so here is the method:

```
quad8( inline('cos(pi*t)'), 0, 5)
```

Likewise, you could call `fplot()` with an inline function definition.

(a) Make a plot of the signal $f(t) = e^t u(-t+1) + e^{2-t}u(t-1)$ over the range $0 \leq t \leq 3$ by using `fplot` and `inline`.

(b) Evaluate the following definite integral of the same function

$$\int_0^3 \Big[e^t u(-t+1) + e^{2-t}u(t-1)\Big]\, dt$$

by using one call to `quad8` and `inline`.

3.4 Using Parameters in the Integrand

One limit of the examples above is that we cannot have any variables in the function definition other than t. However, we want to do Fourier Series calculations, so we know that we must have the parameter k and probably the period T inside the integrals. If you look at the help on `quad8()` you will see that it has additional arguments that can handle these extra parameters. For example, we can construct a general cosine signal with different frequencies by doing the inline definition:

```
inline('cos(2*pi*f*t)','t','f')
```

Even in the case of `inline` we must be careful to identify the names of the different parameters in the function definition. The statement above tells us that t and f are parameters, and gives an ordering: t is the first parameter and f is the second one. This ordering is significant when `fplot` and `quad8` are used because they will treat the first parameter as the variable of interest.

(a) When using `fplot` there are 3 input arguments that are usually skipped[5] in order to pass parameters to the function definition (consult `help fplot`). However, there appears to be a bug in `fplot`, so the "N" argument should be given a value, say 200 to force `fplot` to do a minimum number of evaluations. Here is an example of plotting a chirp, $A\cos(\pi\alpha t^2)$ for `A=100` and `alph=13`.

```
fplot(inline('A*cos(pi*alfa*t.*t)','t','A','alfa'),[0,1],200,[],[],100,13);
```

You must have `N=200` and the two empty matrices in the argument list, and then the number 100 will be used for the amplitude A, and 13 will be used for the value of α.

(b) When using `quad8` there are 2 arguments that must be skipped in order to pass parameters to the function definition (consult `help quad8`). Here is an example of integrating a chirp, $A\cos(\pi\alpha t^2)$.

```
quad8(inline('A*cos(pi*alfa*t.*t)','t','A','alfa'),0,1,[],[],100,13);
```

You must have the two empty matrices in the argument list, so that the last two arguments will be used as parameters, i.e., the number 100 will be used for the amplitude A, and 13 will be used for the value of α. Notice that you must use the "point-star" operator for t^2 because `quad8()` is expecting to evaluate the inline function for a vector of t's.

[5] You can skip arguments by using an empty matrix `[ ]` for each argument to skip.

(c) Follow the style of the previous two parts to make a plot of the signal:

$$x(t) = A\cos(\pi\alpha t^2)\ [u(t) - u(t-1)]$$

over the interval $-1 \leq t \leq 3$. For the parameters, use the values $A = 2.5$ and $\alpha = 10$. In addition, compute the definite integral over the same range.

Instructor Verification (separate page)

4 Exercises: Fourier Series Coefficients

In this lab, the objective is to create a set of functions that will be able to

1. Evaluate the Fourier Series coefficients for the following periodic signal which is defined over one period to be:

$$x(t) = \begin{cases} 120\sin(120\pi t) & \text{for } 0 \leq t \leq 1/120 \\ \\ 0 & \text{for } 1/120 < t < 1/60 \end{cases} \tag{5}$$

   The period is 1/60 seconds. This signal is called a half-wave *rectified* sinusoid, because it contains only the positive lobe of the cosine function. We will use these types of signals and their Fourier Series in the next lab also.

2. Synthesize approximations to $x(t)$ using a finite number of Fourier Series coefficients $\{a_k\}$. This is similar to an earlier exercise in Lab #3, so you might re-use your `fsynthesis` function.

$$x_N(t) = \sum_{k=-N}^{N} a_k e^{j2\pi kt/T_0}$$

   where $2N+1$ is the number of terms used to form the signal and T_0 is the period.

3. Study and explain the convergence of the Fourier Series as $N \to \infty$.

Write down the Fourier Series integral that must be evaluated for the $x(t)$ given in Eq. (5). It is relatively easy to evaluate this integral using the techniques of calculus, but we are going to "do" this integral numerically using MATLAB (see Section 4.4) .

4.1 Function for Fourier Synthesis

In this project we are going to determine Fourier series representations for periodic waveforms, synthesize the signals, and then plot them. In general, the limits on the sum in Eq. (1) are infinite, but for our computational purposes, we must restrict the limits to be a finite number N, which then gives the $2N + 1$ term approximation:

$$x_N(t) = \sum_{k=-N}^{N} a_k e^{j2\pi kt/T_0}. \tag{6}$$

Sometimes it is convenient to define the *fundamental frequency* in rad/s as $\omega_0 = 2\pi/T_0$.

4.1.1 Vectorized Sum of Cosines

This synthesis can be done with the "sum of cosines" function written for Lab #3. That M-file was called `fsynthesis`. The calling sequence of `fsynthesis` is given below. This MATLAB function implements the computation of

$$x(t) = \Re e \left\{ \sum_{k=1}^{L} X_k e^{j2\pi f_k t} \right\} \tag{7}$$

When we use `fsynthesis` for Fourier synthesis, the vector of frequencies will consist of frequencies that are all integer multiplies of the fundamental frequency. In addition, we must include both the positive and negative frequency components, so the input vector of complex amplitudes `Xk` will be a vector of length $L = 2N + 1$ containg the $\{a_k\}$ coefficients in the order $\{a_{-N}, a_{-(N-1)}, \ldots, a_{-1}, a_0, a_1, \ldots, a_N\}$ and the vector `fk` should contain the harmonic frequencies $\{-Nf_0, -(N-1)f_0, \ldots, -f_0, 0, f_0, \ldots, Nf_0\}$ where $f_0 = 1/T_0$ is the fundamental frequency of the periodic signal.

Recall that in Lab #3 you did a synthesis for the case where the Fourier coefficients were given by the formula similar to:

$$a_k = \begin{cases} \dfrac{1}{j\pi k}\left(e^{jk\pi} - 1\right) & k \neq 0 \\ 1 & k = 0 \end{cases} \tag{8}$$

You might want to review those results.

The `fsynthesis` M-file will synthesize a waveform from complex amplitude and frequency information. The first few statements of the M-file are the comment lines that explain the arguments:

```
function        [xx,tt] = fsynthesis(fk, Xk, fs, dur, tstart)
%FSYNTHESIS  VECTORIZED Function to synthesize a sum of cosine waves
%  usage:
%    [xx,tt] = fsynthesis(fk, Xk, fs, dur, tstart)
%      fk = vector of frequencies
%              (these could be negative or positive)
%      Xk = vector of complex amplitudes: Amp*e^(j*phase)
%      fs = the number of samples per second for the time axis
%     dur = total time duration of the signal
%  tstart = starting time (default is zero, if you make this input optional)
%      xx = vector of sinusoidal values
%      tt = vector of times, for the time axis
%
%    Note: fk and Xk must be the same length.
%             Xk(1) corresponds to frequency fk(1),
%             Xk(2) corresponds to frequency fk(2), etc.
```

4.2 Defining a Rectified Cosine

In this part, use the signal definition given in Eq. (5).

(a) Write the auxiliary function that will be needed to define $x(t)$ for use in `fplot()` and `quad8()`. Define the parameters carefully. Turn in the MATLAB code for this function; or if you use the inline method, given the appropriate call to `inline`.

(b) Make a plot of $x(t)$ from Eq. (5) over three periods of the signal: use the range $-1/60 \leq t \leq 2/60$. Hint: you can use MATLAB's `mod()` function to force periodicity in a definition. All you need to

do is replace `t` by `mod(t,T0)` so that you always evaluate the function over the base period of definition. Here is an example that shows the periodicity of a sawtooth wave:

```
fplot('mod(t,3).*(mod(t,3)>=0 & mod(t,3)<=1)',[-5,5],200);
```

with a period of $T_0 = 3$ secs. (The argument 200 is used to force `N=200` or larger in `fplot`.)

(c) Evaluate the DC value of $x(t)$ using numerical integration with `quad8()`. Remember that this is also the a_0 Fourier Series coefficient.

4.3 Fourier Coefficients for a Rectified Sine Wave

Now consider the Fourier Series analysis integral (2).

(a) In this case, the integrand is

$$x(t)e^{-j2\pi kt/T_0}$$

so it is necessary to define another auxiliary function and introduce another parameter, the parameter k. Write this auxiliary function (and turn in the code for it).

(b) Write a MATLAB function that will evaluate the Fourier Series coefficients for the rectified sine wave for $k = -N, \ldots, -1, 0, 1, 2, \ldots, N$. The function will contain a `for` loop to do all the coefficients from $k = -N$ to $k = +N$. It should return a vector containing $2N+1$ elements which are the $\{a_k\}$ coefficients. Turn in the code for this MATLAB function.

(c) Use the auxiliary function written in the previous part to evaluate the Fourier Series coefficients for $x(t)$ from Eq. (5). Find the $\{a_k\}$ coefficients for $-5 \leq k \leq 5$, and make a table of magnitude and phase versus k.

(d) In this part, you must synthesize a set of waveforms using different numbers of Fourier coefficients. Now do the `fsynthesis` synthesis from a finite number of terms to generate $x_N(t)$ for $N = 3$, $N = 7$ and $N = 15$.[6] For each of these synthesized signals make a plot showing the synthesized signal $x_N(t)$ and desired $x(t)$ on the same plot. Use a three-panel subplot to show the three cases on one page.

Reminder: You have to set up a time grid for the time interval, and you should use three periods of the signal from $t = -1/60$ to $t = 2/60$. The spacing between grid points must be small enough so that the sampling rate (grid points per second) is at least *twice as high as the highest frequency component in your signal*, and even more over-sampling will probably be needed to "see" convergence.

(e) Explain how you are getting convergence as N increases. Determine the worst-case approximation error for each value of N in part (d). In other words, determine where the error between the true signal and the Fourier approximation is the largest, and also record the size of that error.

(f) How large do you have to make N so that you cannot see any difference between $x(t)$ and $x_N(t)$ on a plot shown on the computer screen?

4.4 Mathematical Derivation

Derive the mathematical formula for the Fourier Series coefficients for $x(t)$ in Eq. (5). Verify that your formula matches the numerical computation that you did in Section 4.3(c).

[6] In order to use the `fsynthesis()` function for Fourier synthesis, we must include all the $\{a_k\}$ coefficients for both the positive and negative k. Likewise the frequency vector, `fk` must contain both positive and negative harmonics.

Lab #10
ECE-2025
Spring 2001
INSTRUCTOR VERIFICATION SHEET

For each verification, be prepared to explain your answer and respond to other related questions that the lab TA's or professors might ask. Turn this page in at the end of your lab period.

Name: _____ Date of Lab: _____

Lab Quiz #2: Verify that you did SAVE ANSWER for each question, and then submitted the lab quiz with the FINISH button.

Verified:_____ Date/Time:_____

Part 3.2(b) Do the numerical integration of the signal defined over two intervals with `quad8`. Explain what the correct answer should be. Show a mathematical derivation in the space below.

Verified:_____ Date/Time:_____

Part 3.4(c) Use parameter passing in conjunction with the function defined for `fplot` and `quad8`.

Verified:_____ Date/Time:_____