18.03 Problem Set 8

Due by 1:00 P.M., Friday, May 7, 1999, in the boxes at 2-106, next to the Undergraduate Mathematics Office.

Syllabus

IV. The Laplace Transform

31. (F 23 Apr) Basic properties: EP 4.1; handout.

32. (M 26 Apr) Solution of IVPs: EP 4.2 (302–306), 4.3.

33. (W 28 Apr) Discontinuous functions: EP 4.5 (328–333).

34. (F 30 Apr) Convolution and the delta function: EP 4.4 (320–322), 4.6 (341–348).

35. (M 3 May) Transfer functions and Duhamel's principle: Notes LT, EP 4.6 (349–350).

V. Fourier Series and Partial Differential Equations

36. (W 5 May) Fourier series: EP 8.1.

37. (F 7 May) Complex exponentials and Appplications: Handout, EP 8.4.

38. (M 10 May) Heat Equation: EP 8.5.

39. (W 12 May) Wave Equation: EP 8.6 (631–639).

Part I.

34. (F 30 Apr) EP 4.4: 1, 2, 4; 4.6: 1, 2, 3.

35. (M 3 May) EP 4.6: 9, 11.

36. (W 5 May) EP 8.1: 27–29.

Part II.

34. (F 30 Apr) (a) Show that $f'(t) * g(t) - f(t) * g'(t) = f(t)g(0) - f(0)g(t)$ directly from definitions. Then apply the Laplace transform and check that this is compatible with the fact that $\mathcal{L}$ sends convolutions to products and sends $f'(t)$ to $sF(s) - f(0)$.

(b) Compute $e^{at} * e^{bt}$ for $a \neq b$ first by direct integration and then by computing the inverse Laplace transform of $\mathcal{L}(e^{at}; s)\mathcal{L}(e^{bt}; s)$.

(c) A river runs through an agricultural region. It cleanses itself so that pollutants decay as e^{-at}: if there is an amount A at $t = t_0$ then it decays to an amount $Ae^{-a(t-t_0)}$ at time $t \geq t_0$. A new pesticide is introduced into the area at time $t = 0$. It is applied seasonally, so the amount that runs off into the river varies according to the rule (in some units) $x(t) = \sin(2\pi t)$. Express the cumulative amount of this pollutant in the river at time $t \geq 0$ as a convolution and compute it explicitly.

35. (M 3 May) (a) Compute the convolution square $e^{at} * e^{at}$. What is its Laplace transform? At least two other rules governing the Laplace transform lead to the same result; what are they? Explain why the solution to the differential equation $\ddot{x} - 2a\dot{x} + a^2x = r(t)$ with $x(0) = \dot{x}(0) = 0$ is given by the convolution $te^{at} * r(t)$. Take $r(t) = e^{at}$ and find $x(t)$ explicitly.

(b) Express the solution to $x^{(4)} - x = r(t)$ for which the value and first three derivatives of $x(t)$ are all zero at $t = 0$ as an integral using Duhamel's principle.

36. (W 5 May) In this problem you'll use MATLAB to compute the Fourier coefficients of a function. The method will be pretty clumsy but it works reasonably well. The first step will be to produce a vector of values of the function $f(t)$ whose Fourier coefficients we wish to compute. We wish to create a row-vector of say 1000 entries, whose entries are the values of $f(t)$ at a sequence of values of t evenly spaced between $-\pi$ and π. Such a sequence of values of t can be obtained by typing `t=linspace(-pi,pi,1000)`. You should follow this with a semicolon if you want to suppress the screen listing. Let's take for our example the square wave function studied in EP on p. 588. (This won't use the list of values of `t` you may have just created, but other examples and later parts of this problem will.) MATLAB is breathtakingly efficient at creating vectors. Type `y(1:500)=-1` and watch the result. Then type `y(501:1000)=1`. You've done it!

Now create a file called `cf.m` in the directory from which you launched MATLAB (using MATLAB's `edit` if you like). You can skip the comments, in lines following the percent signs, if you like. But notice the new MATLAB commands in use: `length` returns the length of a vector; `sum` sums the entries. Notice the use of `.*`, which causes the give operation to be performed entry by entry on the vector. And we have a loop: the indented operations are performed in sequence for each value of `k` between `k=2` and `k=n+1`.

```
function c=cf(vector,n)

% takes a vector of functional values assumed equally distributed
% across [-pi,pi], and returns a vector of length 2(n+1):
% the sequence of coefficients of cos((k-1)t), for k running
% from 1 to (n+1), in the Fourier expansion, followed by the
% sequence of coefficients of sin((k-1)t), for k running from
% 1 to (n+1).

N=length(vector);
t=linspace(-pi,pi,N);
c(1)=sum(vector)/N;
for k=2:n+1
 c(k)=2*sum(cos((k-1)*t).*vector)/N;
 c(n+1+k)=2*sum(sin((k-1)*t).*vector)/N;
end
```

Your first problem, **(a)**, is to explain what this program does: why does it approximate the integrals defining the Fourier coefficients? (You might want to let the entries of `t` be called t_i, and recall how the integral is approximated by the sum of values of the integrand at these points, times the width of the small intervals.)

Next, create a file `fsum.m` as follows:

```
function y=fsum(c,N)
% takes a vector of Fourier coefficients as created by cf
% and sums the Fourier series at N points equally distributed
% between 0 and 2*pi.
n=length(c)/2-1;
t=linspace(-pi,pi,N);
y=c(1);
for k=1:n
 y=y+c(k+1)*cos(k*t)+c(n+2+k)*sin(k*t);
end
```

Your second problem (**(b)**, easier than the first) is to explain why this does what it claims to do.

Now let's put this to work. Type `c4=cf(y,4)`. You should get a list of 10 numbers. The first 5 should be zero, since they are the coefficients of $\cos(nt)$ and these coefficients are zero since the square wave is an odd function. Next is 0, which is a default value left over from the creation of the vector `c`. Its actual value doesn't matter since it's the coefficient of $\sin(0t) = 0$. Then come the coefficients of $\sin(nt)$. Every other one should be zero, as observed in EP p. 589. Now we'll assemble the Fourier sum: `y4=fsum(c4,1000)`. I chose 1000 points so they would correspond to the values of `t` we computed right at the start. Thus we can plot the resulting function by typing `plot(t,y4)`.

Next, compute the Fourier coefficients through the coefficients of $\cos(16t)$ and $\sin(16t)$: `c16=cf(y,16)`. Check to be sure that the first bunch coincide with what you just computed. Then assemble the corresponding Fourier sum: `y16=fsum(c16,1000)`, and plot it: `hold on`, then `plot(t,y16)`. You might want to add color and a grid. Do the same with 16 replaced by 64 and 256, print out the plot with all four sums, and hand in the result as **(c)**. You should come pretty close to the square wave, except near the points of discontinuity.

The final thing to check, in this problem, is the "Gibbs phenomenon": near the jump point at $t = 0$ and the one at $t = \pm\pi$, the Fourier approximations seem to overshoot, no matter how many terms you take. You can see this best by killing the current Figure and starting a new one showing just one sum: `plot(t,y64)`, for example. You can find the largest value of the 1000 values we have by typing `max(y64)`. Part **(d)** is the answer. This gives an estimate of the overshoot. (If you try `max(y256)` you'll get a smaller number. This is misleading, though, and it occurs because the overshoot occurs in such a small range that it is missed by the mesh of 1000 points we are polling at.) The fact is that as the number of terms in the Fourier expansion gets large, the maximal value of the sum converges not to 1 but to

$$\frac{2}{\pi}\int_0^\pi \frac{\sin(t)}{t}dt \simeq 1.17897974447217.$$

(e) Use `cf.m` to compute the Fourier coefficients of the saw-tooth wave $f(t) = t, -\pi < t < \pi$, up to the coefficients of $\sin(8t)$. Why are the coefficients of $\cos(nt)$ all zero?