

Fourier Transforms

Introduction

Up to now, we've looked at techniques to find voltages and currents in circuits

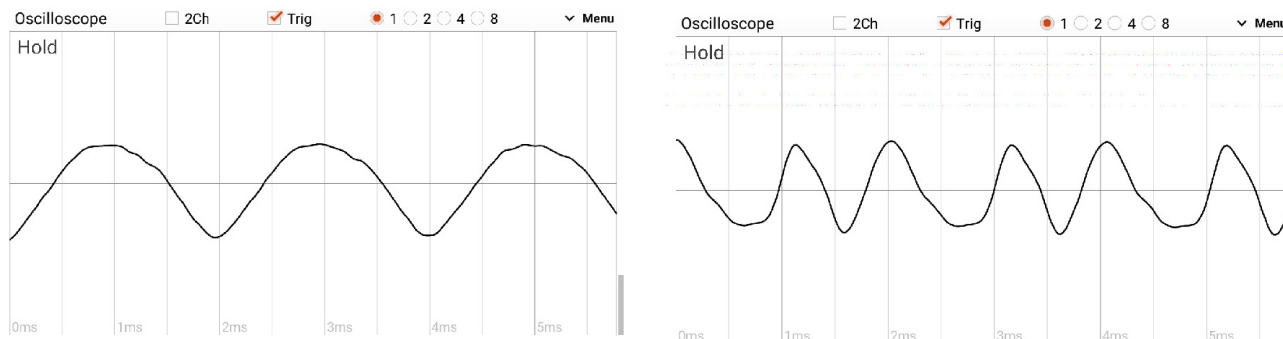
- With DC inputs,
- With AC inputs (phasors),
- With multiple inputs (superposition).

What happens if the input is

- Periodic in time T , but
- Not a sine wave?

This is actually a very common problem. Many signals are periodic in nature, but not a sine wave. For example, different musical instruments playing the same note sound different. The period is the same (it's the same note), but the waveforms are different. For example, Figures 1 show the waveforms for the note A4 (500Hz) for a piano and a cello.

How do you find the voltages and currents in an RLC circuit, such as that shown in Figure 2, when the input



waveform is that of a piano? Of a cello?

Figure 1: Waveform for playing note A4 (500Hz) on a piano (left) and a cello (right)

A second example comes from CircuitLab. With CircuitLab, the AC inputs can be

- Sine waves
- Triangle waves, or
- Sawtooth waves

How do you find the output of an RLC circuit such as that shown in Figure 2 when the input is a square wave?

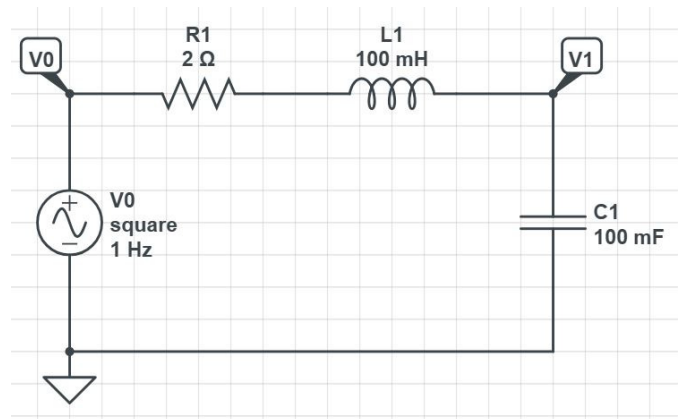


Figure 2: RLC circuit with a periodic input (V0) that's not a sine wave

One, solution of course, is to simulate that circuit using CircuitLab. For the circuit of Figure 2 with a square wave input, the result of a time-domain simulation is shown in Figure 3.

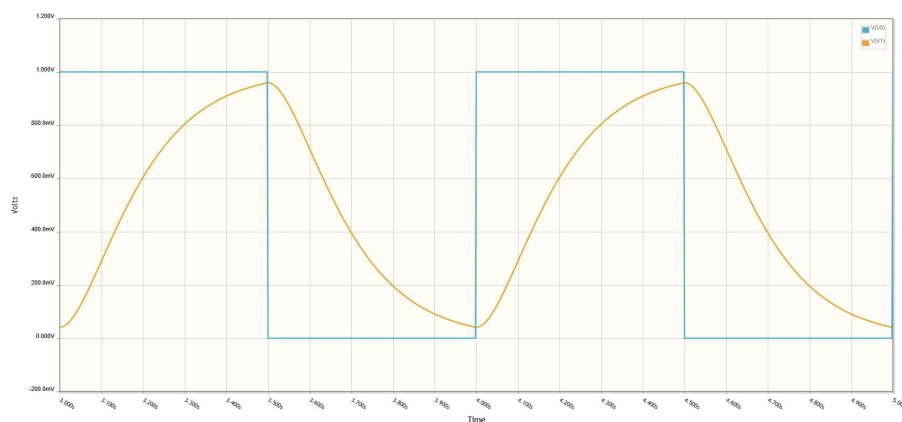


Figure 3: Time-domain simulation of the RLC circuit showing V0 (blue) and V1 (orange).

While CircuitLab is able to show you the shape of the output (assuming you can get the input into CircuitLab), it doesn't tell you the equation for $V1(t)$.

In this lecture, a procedure is given for finding voltages and currents for circuits with periodic inputs. This procedure uses a technique called *Fourier Transforms*.

Fourier Transforms are normally covered in Signals and Systems (two courses after Circuits I). When you take Signals, you'll spend around five weeks on this topic. We're covering it here since

- Once you have phasors and
- Superposition

you've got Fourier transforms. At this point, we should be ready for it.

Fourier Transforms

Fourier Transforms state that if you have a signal that's periodic in time T

$$x(t) = x(t + T)$$

then you can express $x(t)$ as an infinite series

$$x(t) = \sum_{n=0}^{\infty} a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)$$

where

$$\omega_0 = \frac{2\pi}{T}.$$

What this means is as follows.

- ω_0 is the fundamental frequency in rad/sec. For example, the note A4 (500Hz) is periodic in 2.00ms. The harmonics ($n=1..infinity$) are also periodic in time T.
- Going right to left. If you add together a bunch of sine waves that are periodic in time Tm the result is also periodic in time T. That's kind of a no-brainer.
- Going left to right. This is where Fourier transform gets interesting. If $x(t)$ is periodic in time T, it is composed of a bunch of sine waves that are harmonics of the fundamental (500Hz here).

Fourier Transforms: Right to Left

Going right to left, Fourier transforms simply state that if you add functions that are periodic in time T, the result is also periodic in time T. To get different waveforms, add up the harmonics with different weightings.

If you go to the CRC Handbook of Mathematics, several pages of Fourier Transforms are given. A few of these are:

- Square Wave:
$$x(t) = \frac{4}{\pi} \sum_{n \text{ odd}} \left(\frac{1}{n}\right) \sin(nt)$$
- Triangle Wave:
$$x(t) = \frac{8}{\pi^2} \sum_{n \text{ odd}} \left(\frac{1}{n^2}\right) \cos(nt)$$
- Sawtooth Wave:
$$x(t) = \frac{2}{\pi} \sum_n (-1)^{n-1} \left(\frac{1}{n}\right) \sin(nt)$$
- Half-Rectified Sine Wave:
$$x(t) = \frac{1}{\pi} + \frac{1}{2} \sin(x) - \frac{2}{\pi} \sum_{n \text{ even}} \frac{1}{n^2-1} \cos(nt)$$
- Full-Rectified Sine Wave:
$$x(t) = \frac{2}{\pi} - \frac{4}{\pi} \sum_{n \text{ even}} \frac{1}{n^2-1} \cos(nt)$$

Using Matlab, these can be verified.

Square Wave: For example, the Fourier transform for a square wave with a period of 2π is

$$x(t) = 0.5 + \sum_{n \text{ odd}} \frac{2}{\pi n} \sin(nt)$$

Matlab code to implement this as n goes from 1 to 39 is as follows:

```
t = [0:0.001:2]' * 2*pi;
y = 0.5 + 0*t;

for i=1:20
    n = 2*i-1;
    y = y + 2/(n*pi) * sin(n*t);
    plot(t,y);
    title(n);
    pause(1);
end
```

The plot of $x(t)$ out to the 9th harmonic (red) and infinite (blue) is shown in Figure 4.

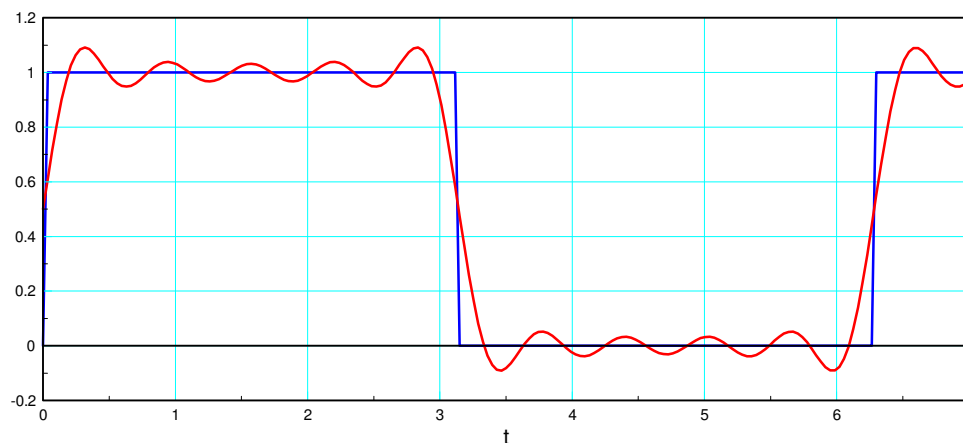


Figure 4: Fourier series approximation to a square wave taken out to the 9th harmonic (red) and infinite (blue)

Note that

- Each term is periodic in 2π seconds. The sum is also periodic in 2π seconds.
- With this set of weightings, the sum converges to a square wave

It could take a lot of terms. Eventually, the series converges to a square wave.

Triangle Wave: The Fourier transform for triangle wave is

$$x(t) = \frac{8}{\pi^2} \sum_{n \text{ odd}} \left(\frac{1}{n^2} \right) \cos(nt)$$

Matlab code to implement this as n goes from 1 to 39 is as follows:

```
t = [0:0.001:2]' * 2*pi;
y = 0*t;

for i=1:20
    n = 2*i-1;
    y = y + 8/pi^2 * (1/(n^2)) * cos(n*t);
    plot(t,y);
    title(n);
    pause(1);
end
```

The waveform out to the 9th harmonic and infinite are shown in Figure 5.

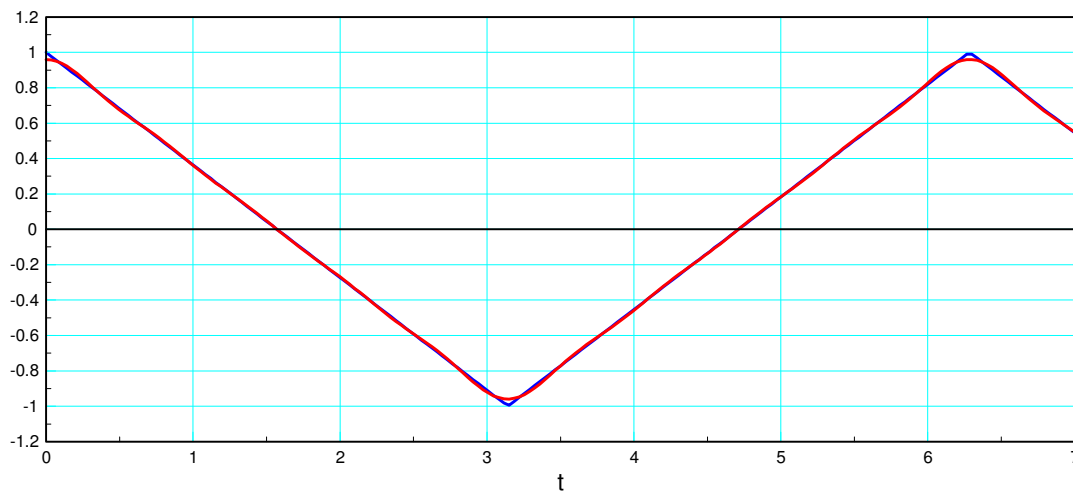


Figure 5: Fourier Series approximation to a triangle wave out to the 9th harmonic (red) and infinite (blue)

Again, note that

- Each term is periodic in 2π seconds. The sum is also periodic in 2π seconds.
- With this set of weightings, the sum converges to a triangle wave

This converges faster since the high frequency terms drop off as $1/n^2$ rather than $1/n$ for a square wave. If you change the code, you can also approximate the other waveforms.

Fourier Transform: Going Left to Right

The more interesting direction is going left to right. Given a waveform that's periodic in time T , you can express it in terms of sine waves at the fundamental frequency and its harmonics:

$$x(t) = \sum_{n=0}^{\infty} a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)$$

So here's the question: how do you determine the terms a_n and b_n ?

Method #1: Curve Fitting: There are a couple of ways of doing this. One method is to use least squares curve fitting.

With least squares, you can curve fit any function. The procedure is to

- First, define the basis function.
- Then solve for the weightings using least squares

For example, assume $x(t)$ is a square wave with period 2π :

$$T = 2\pi$$

$$x(t) = x(t + T)$$

$$x(t) = \begin{cases} +1 & 0 < t < \pi \\ 0 & \pi < t < 2\pi \end{cases}$$

$x(t)$ can be approximated with a power series:

$$x(t) \approx a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4$$

Here, the basis function is

$$B = [1, t, t^2, t^3, t^4]$$

The coefficients can be found using least squares in Matlab

```
>> dt = 0.0001;
>> T = 2*pi;
>> t = [0:dt:T]';
>> x = 1*(t < (T/2));
>> B = [t.^0, t, t.^2, t.^3, t.^4];
>> A = inv(B'*B)*B'*x

a0    0.8125
a1    0.5968
a2   -0.3325
a3    0.0353
a4    0.0000

>> plot(t, x, 'b', t, B*A, 'r')
```

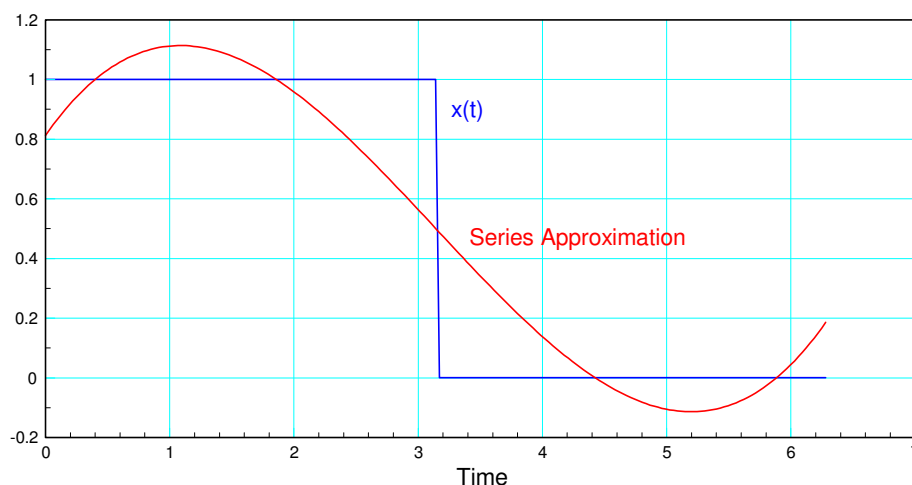


Figure 6: Power series approximation to a square wave

As you add more and more terms, the resulting curve fit gets better and better. The result, however, isn't useful since the basis isn't made up of sine waves. It likewise doesn't help with phasor analysis.

Instead of using a power series, let the basis be a bunch of sine waves:

$$x(t) = a_0 + a_1 \cos(t) + b_1 \sin(t) + a_2 \cos(2t) + b_2 \sin(2t) + a_3 \cos(3t) + b_3 \sin(3t)$$

Using least squares, each of the coefficients can be found like before. In Matlab:

```
>> T = 2*pi;
>> t = [0:0.001:T]';
>> x = 1*(t<T/2);
>> B = [t.^0, cos(t), sin(t), cos(2*t), sin(2*t), cos(3*t), sin(3*t)];
>> A = inv(B'*B)*B'*x

a0    0.5000
a1    0.0001
b1    0.6366
a2   -0.0000
b2   -0.0000
a3    0.0001
b3    0.2122

>> plot(t,x,'b',t,B*A,'r')
```

The resulting curve fit is shown in Figure 7. Once again, as you add more and more terms, the curve fit gets better and better. Unlike with the power series, however, this curve fit *is* useful. $x(t)$ is now expressed in terms of sine waves.

$$x(t) = 0.5 + 0.6366 \sin(t) + 0.2122 \sin(3t)$$

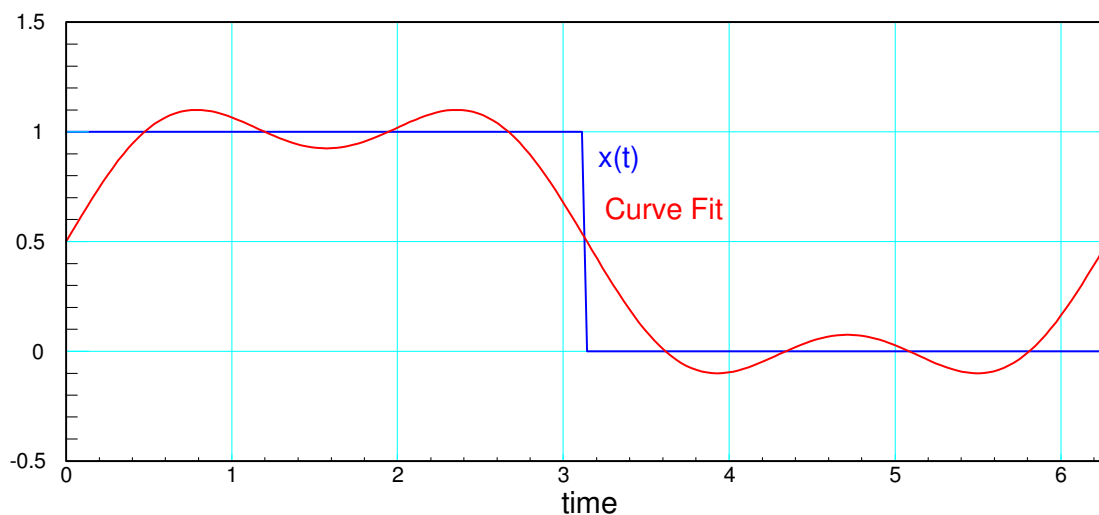


Figure 7: $x(t)$ and it's approximation using a basis consisting of sine waves

Method #2: Fourier Transforms

Another way to compute the coefficients is to note that this is an orthogonal basis:

$$x(t) = a_0 + a_1 \cos(t) + b_1 \sin(t) + a_2 \cos(2t) + b_2 \sin(2t) + a_3 \cos(3t) + b_3 \sin(3t)$$

What this means is that average of basis term multiplied with another is zero:

$$\int_T \cos(n\omega_0 t) \cdot \sin(m\omega_0 t) \cdot dt = 0$$

$$\int_T \cos(n\omega_0 t) \cdot \cos(m\omega_0 t) \cdot dt = 0 \quad n \neq m$$

$$\int_T \sin(n\omega_0 t) \cdot \sin(m\omega_0 t) \cdot dt = 0 \quad n \neq m$$

It is *not* an orthonormal basis, however

$$\int_T 1 \cdot 1 \cdot dt = T$$

$$\int_T \cos(n\omega_0 t) \cdot \cos(n\omega_0 t) dt = \frac{T}{2}$$

$$\int_T \sin(n\omega_0 t) \cdot \sin(n\omega_0 t) dt = \frac{T}{2}$$

This allows you to pull out each term in $x(t)$ separately:

$$a_0 = \frac{1}{T} \int_T x(t) \cdot 1 \cdot dt$$

$$a_n = \frac{1}{2T} \int_T x(t) \cdot \cos(nt) \cdot dt$$

$$b_n = \frac{1}{2T} \int_T x(t) \cdot \sin(nt) \cdot dt$$

In Matlab:

```
>> T = 2*pi;
>> t = [0:0.001:T]';
>> x = 1*(t<T/2);
>> a0 = mean(x)
a0 = 0.5000

>> a1 = 2 * mean(x .* cos(t))
a1 = 1.8862e-004

>> b1 = 2 * mean(x .* sin(t))
b1 = 0.6365

>> a2 = 2 * mean(x .* cos(2*t))
a2 = 1.2965e-004

>> b2 = 2 * mean(x .* sin(2*t))
b2 = -7.6835e-008

>> a3 = 2 * mean(x .* cos(3*t))
a3 = 1.8862e-004

>> b3 = 2 * mean(x .* sin(3*t))
b3 = 0.2122
```

Note that these are the same results obtained through curve fitting. That's really all Fourier transforms are: they're a curve fit for a periodic function using a basis that's composed of sine waves.

Circuit Analysis Using Fourier Transforms

Fourier transforms can be used to find voltages in a circuit where the input is a periodic function that's *not* a sine wave. For example, consider the circuit of Figure 8. Using phasors and voltage division,

$$V_1 = \left(\frac{1}{LC(j\omega)^2 + RC(j\omega) + 1} \right) V_0$$

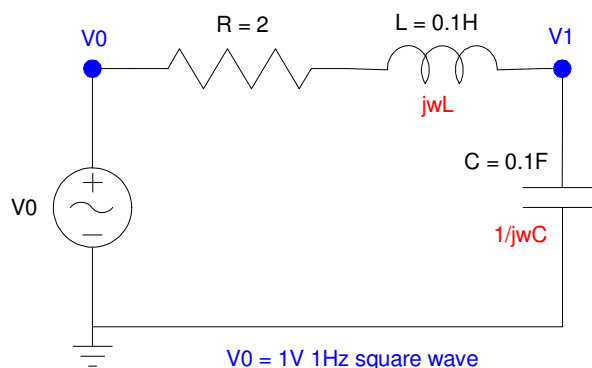
Find V_1 when V_0 is a 1V, 1Hz square wave:

V_0 is a 1Hz periodic signal

$$v_0(t) = v_0(t + 1)$$

V_0 is a 50% duty cycle square wave

$$v_0(t) = \begin{cases} 1 & 0 < t < 0.5 \\ 0 & 0.5 < t < 1 \end{cases}$$

Figure 8: Find $V_1(t)$ when $V_0(t)$ is a 1Hz square wave

Procedure: Start with expressing $V_0(t)$ in terms of its Fourier Series. Going out to the 3rd harmonic:

$$V_0(t) \approx 0.5 + 0.6365 \sin(2\pi t) + 0.2122 \sin(6\pi t)$$

V_0 contains three different frequencies. Use superposition to treat this like three separate problems.

DC Analysis: Assume

$$V_0(t) = 0.5$$

$$V_1 = \left(\frac{1}{LC(j\omega)^2 + RC(j\omega) + 1} \right)_{\omega=0} V_0$$

$$V_1 = 0.5$$

1 Hz Analysis:

$$v_0(t) = 0.6365 \sin(2\pi t)$$

In phasor form

$$\omega = 2\pi$$

$$V_0 = 0 - j0.6365$$

$$V_2 = \left(\frac{2}{(j0.04\omega)^2 + (j0.04\omega) + 1} \right)_{\omega=2\pi} (-j0.6365)$$

$$V_2 = -0.3397 - j1.2678$$

meaning

$$v_2(t) = -0.340 \cos(2\pi t) + 1.268 \sin(2\pi t)$$

2 Hz Analysis:

There is no input at 2Hz. Hence, there is no output at 2Hz.

3 Hz Analysis

$$v_0(t) = 0.2122 \sin(6\pi t)$$

In phasor form

$$V_0 = 0 - j0.2122$$

$$\omega = 6\pi$$

V1 is then

$$V_1 = \left(\frac{2}{(j0.04\omega)^2 + (j0.04\omega) + 1} \right)_{\omega=6\pi} (-j0.2122)$$

$$V_1 = -0.0386 + j0.0261$$

meaning

$$v_1(t) = -0.0386 \cos(6\pi t) - 0.0261 \sin(6\pi t)$$

Putting it all together

$$v_2(t) = 0.5 - 0.340 \cos(2\pi t) + 1.268 \sin(2\pi t) \\ - 0.0386 \cos(6\pi t) - 0.0261 \sin(6\pi t)$$

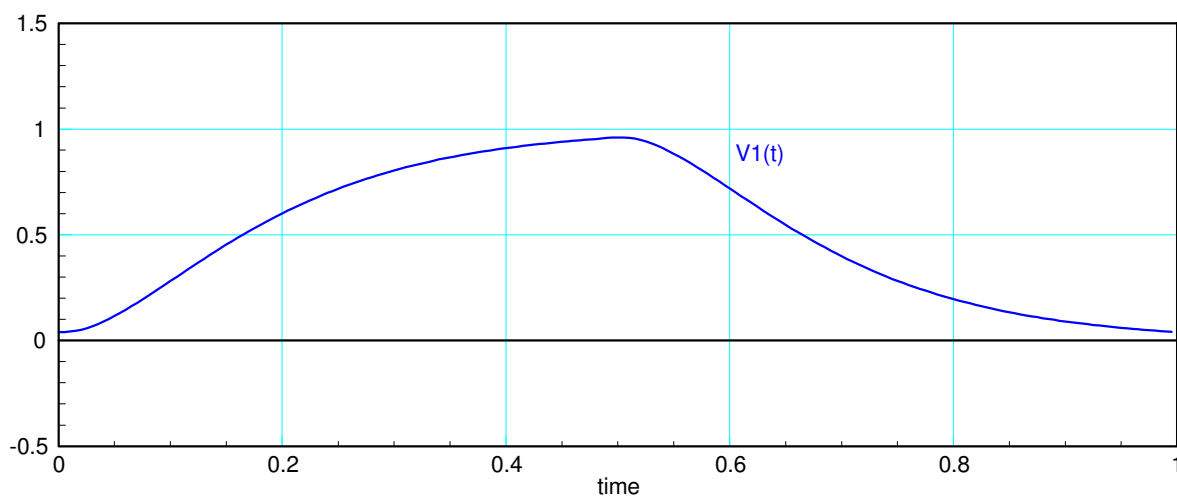


Figure 9: $V_1(t)$ computed using the Fourier series approximation to $V_0(t)$

Summary:

Once you have phasors and superposition, you have Fourier Transforms.

Fourier Transforms allow you to solve differential equations with *any* periodic input

- Express $x(t)$ as the sum of a DC term and harmonics
- Find $y(t)$ at each frequency separately
- Add the results together to get the total $y(t)$

Matlab makes this process fairly easy. It's a little tedious if there are lots of terms in the answer, but it gets you there. The power of Fourier transforms is it lets you find the voltages and currents for a circuit with *any* periodic input.