

CE2004: Circuits & Signal Analysis Part 2

Lu Shengliang

SLU001

School of Computer Engineering

Nanyang Technological University

SLU001@e.ntu.edu.sg

October 29, 2013

Abstract

lab 4: Use *Scilab* software package to demonstrate how signals can be simulated. *Step*, *Rectangular*, *Sinusoidal*, *delta* functions and *Square* series were conducted.

1 Defining and Plotting Step Functions

define a step function called *step(t)* that is equal to 1 when $t \geq 0$, and is equal to 0 when $t < 0$.

1.1 Step function $u(t)$

```
function y = step(t)
    y = round((sign(t) + 1) / 2)
endfunction
```

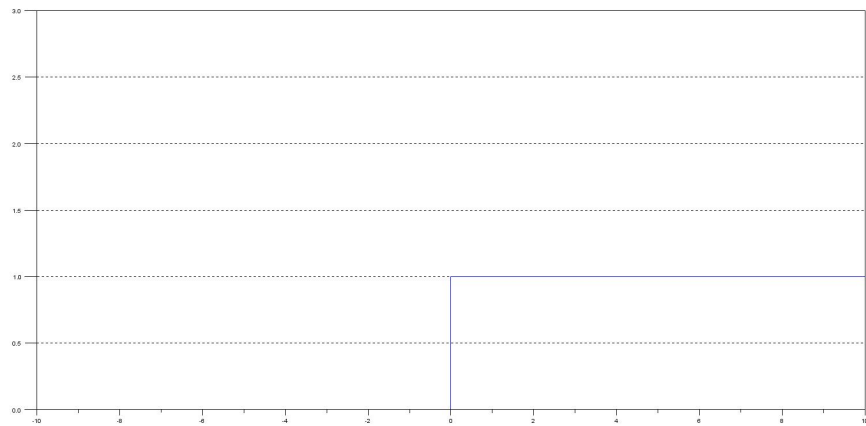


Figure 1: `stepfunctionu(t)`

1.2 Plotting of different step functions

```
plot(t, step(t - 1))
```

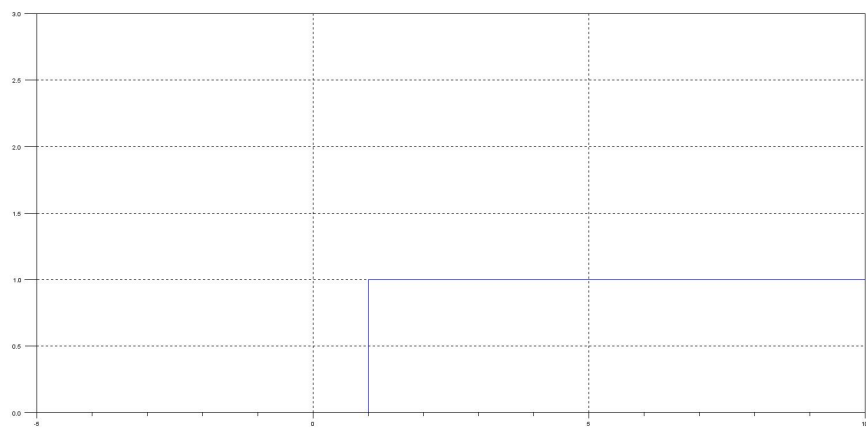


Figure 2: $u(t - 1)$

```
plot(t, 2 * u(t + 2))
```

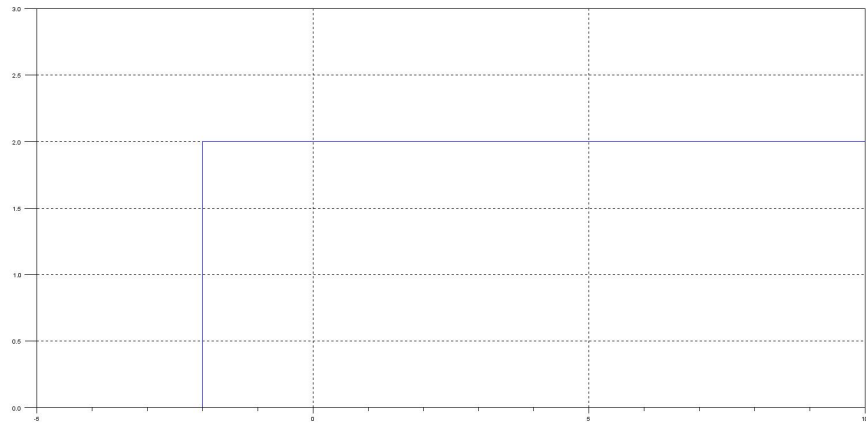


Figure 3: $2 * u(t + 2)$

```
plot(t, u(t + 1) - u(t - 1))
```

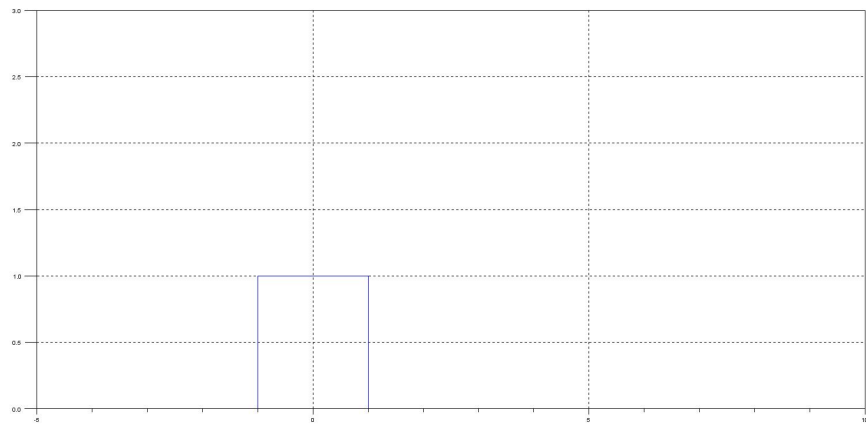


Figure 4: $u(t + 1) - u(t - 1)$

1.3 Question 5.1

Question:

Save the plots and comment on the observed shapes of the signals.

Answer:

The step function was generated by built-in *sign()* function. It returns 1 when $t \geq 0$, otherwise 0. So $u(t - 1)$ is basically step function after shifting right 1 unit on x-axis. $2 * u(t + 2)$ is step function shifting 2 unit left and doubling the height of sign. $u(t + 1) - u(t - 1)$ is actually the way of generating Π function.

2 Defining and Plotting Rectangular Functions

that is equal to 1 when $-0.5 \leq t \leq 0.5$ and is equal to 0 otherwise. Use the previously defined step function to define a square function $\Pi(t)$.

2.1 Square function $\pi(t)$

```
function y = pi(t)
    y = step(t + 0.5) - step(t - 0.5)
endfunction
```

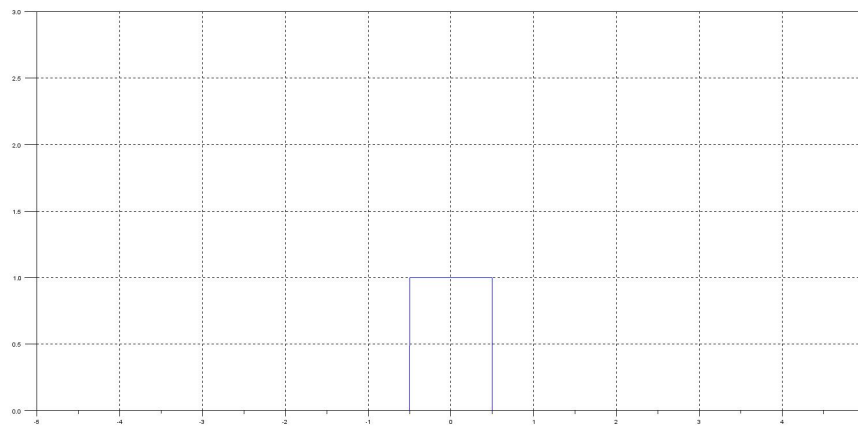


Figure 5: squarefunction $\Pi(t)$

```
plot(t, pi(2 * t - 1))
```

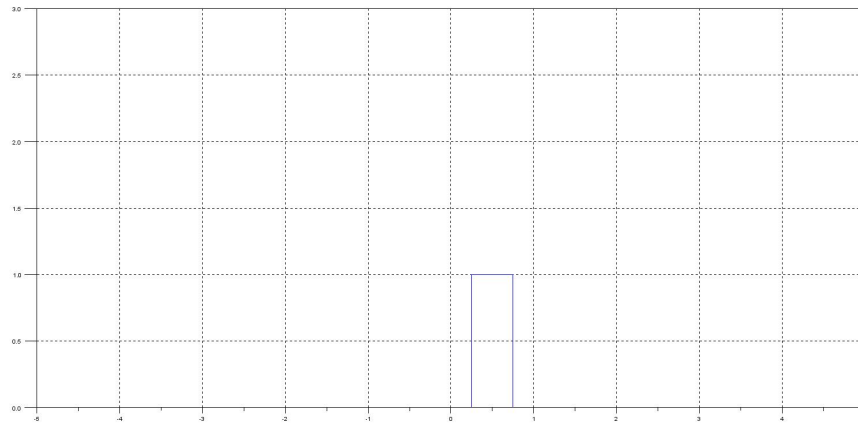


Figure 6: $\Pi(2t - 1)$

```
plot(t, 1.5 * pi(-6 * t + 5))
```

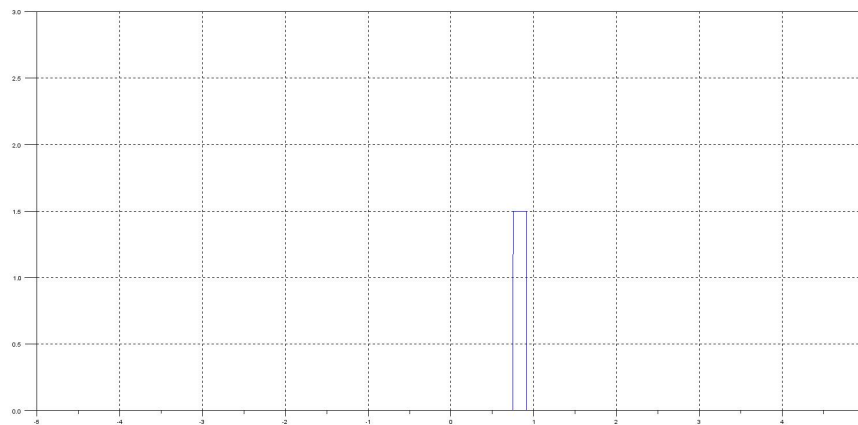


Figure 7: $1.5 * \Pi(-6t + 5)$

```
function y = Pic(t)
    y = Pi(0.5*t-2) + 2*Pi(0.5*t-1) + 3*Pi(0.5*t)
        + 2*Pi(0.5*t+1) + Pi(0.5*t+2)
```

```
endfunction
```

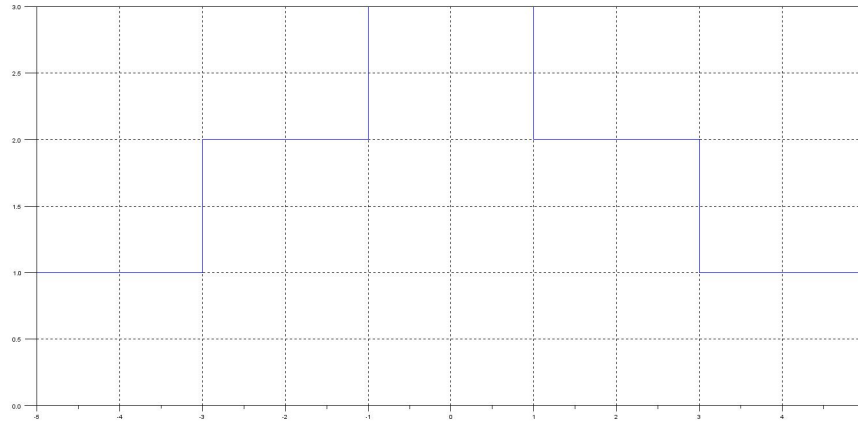


Figure 8: $\Pi(0.5t - 2) + 2\Pi(0.5t - 1) + 3\Pi(0.5t) - 2\Pi(0.5t + 1) + \Pi(0.5t + 2)$

2.2 Question 5.2

Question:

Save the plots and comment on the observed shapes of the signals.

Answer:

The square function was generated by *step()* function. It returns 1 when when $-0.5 \leq t \leq 0.5$ and is equal to 0 otherwise. So $\Pi(2t - 1)$ is basically square function shifting right 1 unit on x-axis and compressing a half on x-axis. $1.5 * \Pi(-6t + 5)$ is square function shifting right $\frac{5}{6}$ unit on x-axis and compressing by 6 then increasing the height of sign to 1.5 times. The last one is actually the combination of 5 different heights Π function.

3 Defining Sinusoidal Signals and Converting Them into Sounds

Use built-in sine function *sin(t)* to generate the following signals with 0.0001 sampling interval and save them using *savewave()* command.

3.1 2-second duration middle C

2-seconds duration signal corresponding to a sine wave with frequency 261.63Hz

```
function y = wavA(t)
    y = sin(261.63*2*%pi*t)
endfunction

t = 0 : 0.0001 : 4.41
```

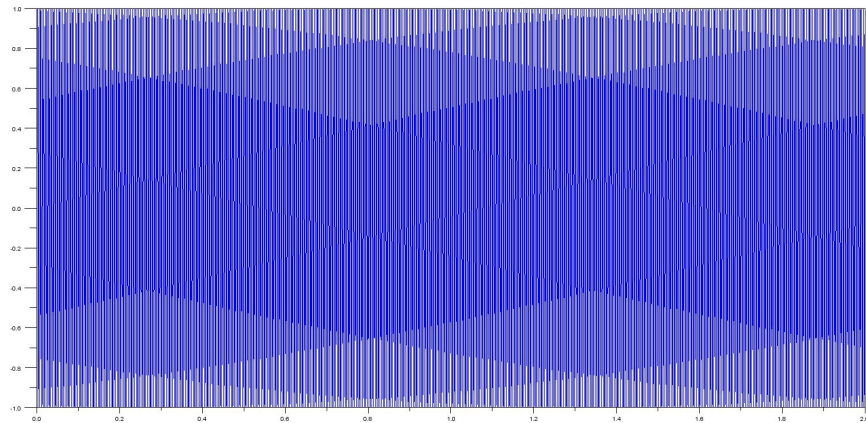


Figure 9: middle C

Because 22050 is the default value of number of sample per second, in order to get a 2-second duration sound, the length of t should be 4.41.

$$\text{rangeT} = 22050 \text{ sample/second} \times 2 \text{ second} \times 0.0001 \text{ sample}^{-1} \quad (1)$$

$$\text{rangeT} = 4.41 \quad (2)$$

3.2 2-second duration composited wave

```
function y = sumOfWav(t)
```

```

y = sin(261.63*2*%pi*t)
  + sin(329.63*2*%pi*t)
  + sin(392.00*2*%pi*t)
endfunction

```

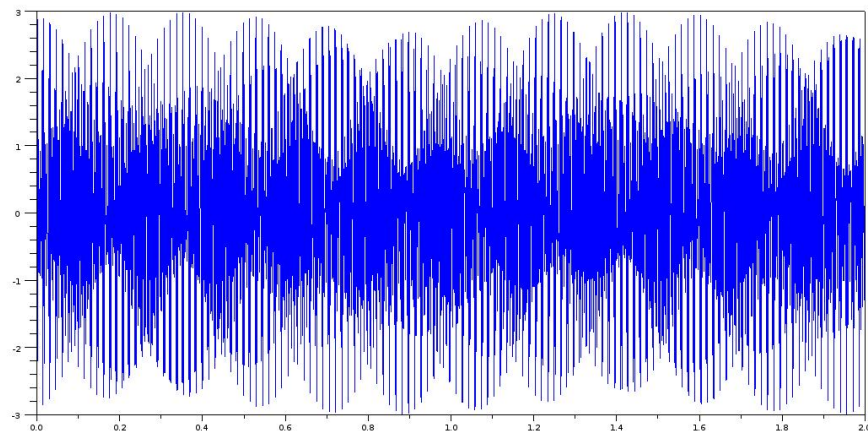


Figure 10: Sum of Wave

3.3 6-second duration consecutive C, E, G notes

```

function y = sixSecond(t)
    y = (sin(261.63*2*%pi*t) .* Pi(1/4.41*t-0.5))
      + (sin(329.63*2*%pi*t) .* Pi(1/4.41*t-1.5))
      + (sin(392.00*2*%pi*t) .* Pi(1/4.41*t-2.5))
endfunction

t = 0 : 0.0001 : 13.23

```


$$\text{rangeT} = 22050 \text{sample/second} \times 6 \text{second} \times 0.0001 \text{sample}^{-1} \quad (3)$$

$$\text{rangeT} = 13.23 \quad (4)$$

If sample interval is reduces to 0.01:

```
function y= sixSecond2(t)
    y = (sin(261.63*2*%pi*t) .* Pi(1/441*t-0.5))
        + (sin(329.63*2*%pi*t) .* Pi(1/441*t-1.5))
        + (sin(392.00*2*%pi*t) .* Pi(1/441*t-2.5))
endfunction

t = 0 : 0.01 : 1323
```

$$\text{rangeT} = 22050 \text{sample/second} \times 6 \text{second} \times 0.01 \text{sample}^{-1} \quad (5)$$

$$\text{rangeT} = 1323 \quad (6)$$

3.4 Question 5.3

Question 1:

Play back the saved wave files using Windows Media player and comment on what you hear.

Answer 1:

The sound is simple and complete for middle C, sounds like phone busy sound. The composited sound is noisy and sounds like *BU-ZZZ*. The 6-second duration sound like the output of simple music player or simple buzzer, with 3 different tone. If sample interval is reduces to 0.01, the sound becomes sharper than before and the tones are decreased instead of increased.

Question 2:

Why do we need to use such a small sampling interval? What would happen if sampling interval is reduced to 0.01?

Answer 2:

Since the vector t we are using is discrete, the wave we generated is basically the assembled dots connected with smooth wave lines. The more concentrated as the dot allocated, the more real the wave sounds. If using 0.01 as first sampling interval and using 0.0001 as second sampling interval, since we will generate two 6-second waves, the first one with wider sampling interval with return a higher frequency wave. A higher frequency wave sounds far differently from real sound and, because of higher frequency, the sound will have higher tones.

4 Defining Delta Function and Verifying Its Properties

Take use of the following method:

$$a\Pi(at) \xrightarrow{a \rightarrow \infty} \delta(t) \quad (7)$$

4.1 $\delta(t)$ and $e^t\delta(t-1)$

```
function y = delta(t)
    y = 200 * Pi(200 * t);
endfunction

function y = expDelta(t)
    y = %e^t .* delta(t - 1)
endfunction
```

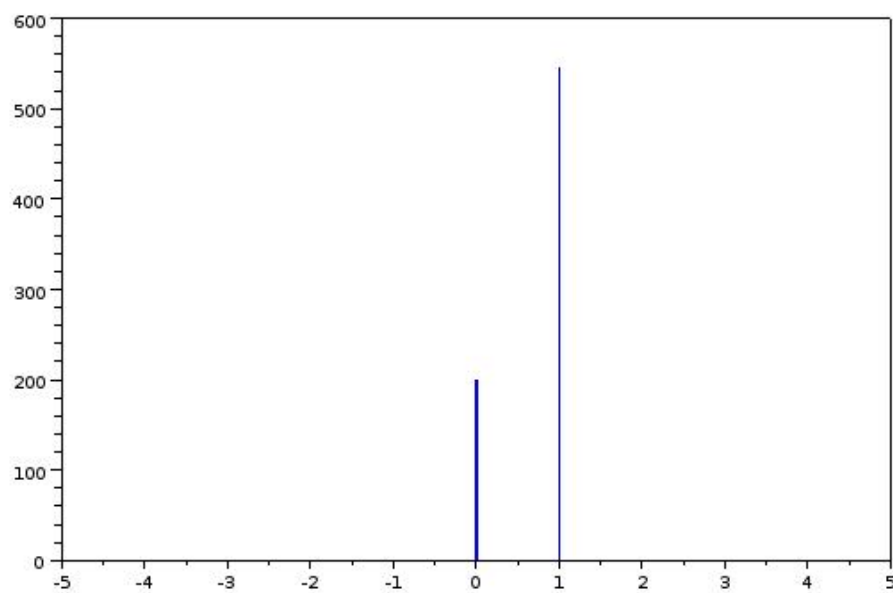
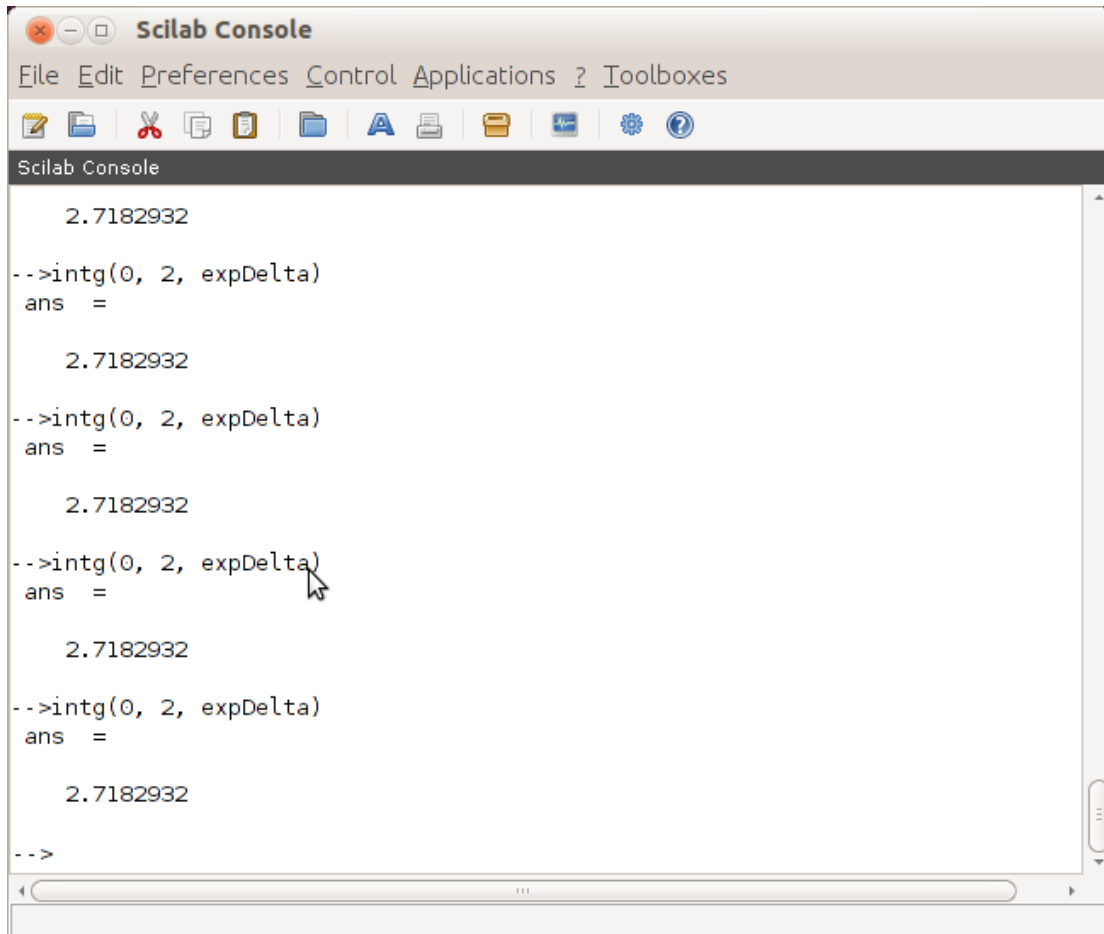


Figure 11: $\delta(t)$ and $e^t \delta(t - 1)$ (when $\alpha = 200$)

4.2 Evaluate $\int_0^2 e^t \delta(t-1) dt$

A screenshot of the Scilab Console window. The window has a title bar with standard OS controls and a menu bar with options: File, Edit, Preferences, Control, Applications, and Toolboxes. Below the menu bar is a toolbar with icons for file operations and editing. The main area of the window is a text editor showing the following text:

```
2.7182932
-->intg(0, 2, expDelta)
ans =
2.7182932
-->intg(0, 2, expDelta)
ans =
2.7182932
-->intg(0, 2, expDelta)
ans =
2.7182932
-->intg(0, 2, expDelta)
ans =
2.7182932
-->
```

A mouse cursor is visible over the text 'expDelta' in the fourth line of the code block.

Figure 12: $\int_0^2 e^t \delta(t-1) dt$ result

4.3 Question 5.4

Question:

Save the plots and comment on the observed shapes and integration results using sampling and shifting property of the delta function.

Answer:

The value of a is not supposed to be very large, otherwise, the value will be ignored due to precision issue.

The $e^t \delta(t-1)$ was generated by $\delta(t)$ through 1-unit right shifting and then lengthen its height to e times than the original delta function.

Because the value of $\delta(t - 1)$ equals to ∞ only when $t = 1$, otherwise 0; the integration $\int_0^2 e^t \delta(t - 1) dt$ will only return the value of e^t at $t = 1$. So the result is basically e .

5 Signal Composition

Define a function called `squareseries` that is equal to $\sum_{k=1, k-\text{odd}}^n \frac{1}{k} \sin(kt)$.

```
function y = kSumSin(k, t)
    y = 1/k * sin(k*t)
endfunction

function y = square_series(n, t)
    y = 0:0:0
    for i = 1 : 2 : n
        y = y + kSumSin(i, t)
    end
endfunction
```

5.1 $n = 1$

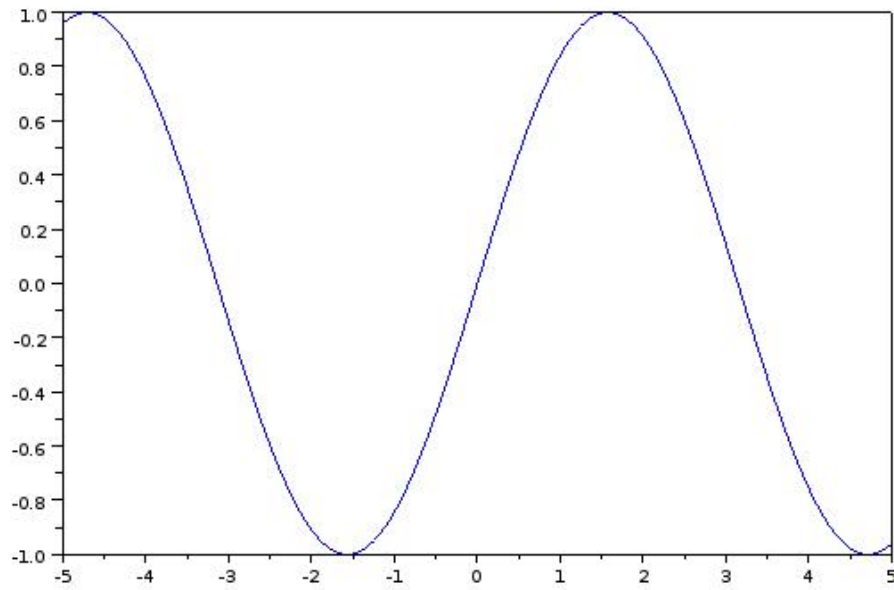


Figure 13: $\sin(t)$

5.2 $n = 3$

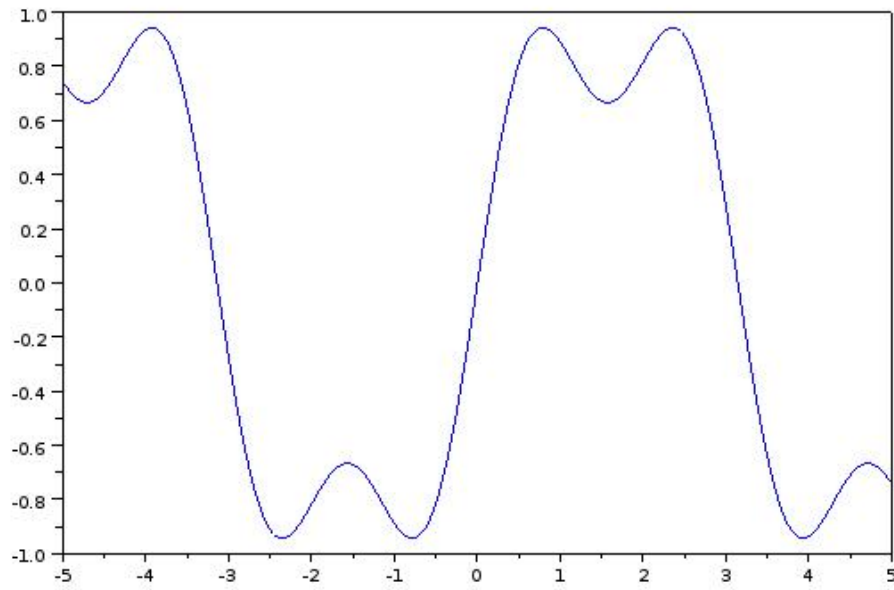


Figure 14: $\sum_{k=1,3} \frac{1}{k} \sin(kt)$

5.3 $n = 9$

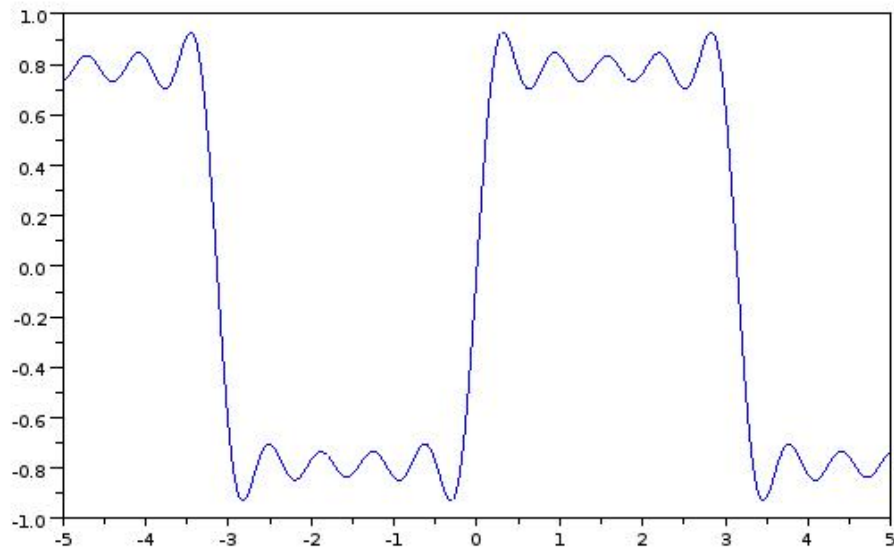


Figure 15: $\sum_{k=1,3,5,7,9} \frac{1}{k} \sin(kt)$

5.4 $n = 100$

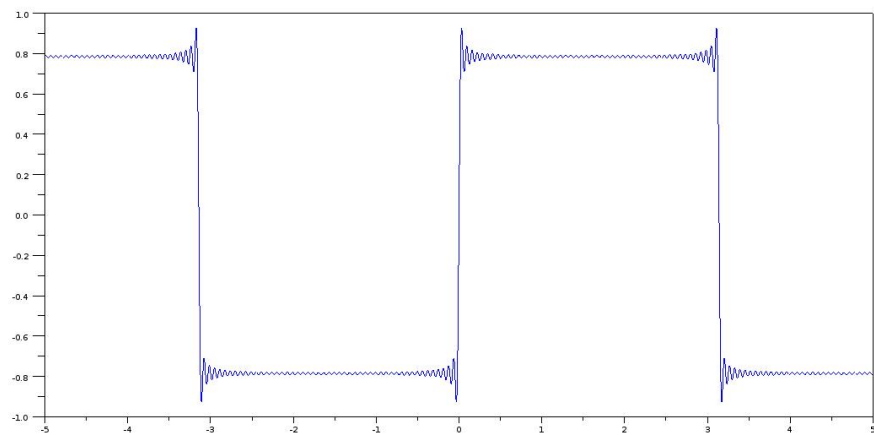


Figure 16: $\sum_{k=1, k-\text{odd}}^{100} \frac{1}{k} \sin(kt)$

5.5 Question 5.5

Question:

Save the plots and comment on the shapes of observed signals.

Answer:

A set of sine waves are supposed to be able to generate a square wave according to trigonometric series:

$$\Pi(t) = \sin\omega t + \frac{1}{3}\sin 3\omega t + \frac{1}{5}\sin 5\omega t + \dots \quad (8)$$

When $n = 1$, the output is basically $\sin(t)$.

When $n = 3, 9$, the wave is an abstract square wave with unstable data at certain times.

When $n = 100$, the shape of square is already given, with unstable signal at both of the rising and dropping edges.

As n goes bigger, the wave will be more accurate, which means it will look and perform more similar to a square wave.