The Structure and Interpretation of Signals and Systems
Solutions Manual

Edward A. Lee and Pravin Varaiya
eal@eecs.berkeley.edu, varaiya@eecs.berkeley.edu
Electrical Engineering & Computer Science
University of California, Berkeley

August 2002

Contents

Chapter 1

Signals

1. (a) $x \in [\textit{Integers} \rightarrow \textit{Reals}]$ given by

$$\forall n \in \textit{Integers}, \quad x(n) = n$$

(b) $x \in [\textit{Reals} \rightarrow \textit{Reals}^2]$ given by

$$\forall t \in \textit{Reals}, \quad x(t) = (t, 2t)$$

(c) $x \in [\{0, 1, \cdots, 600\} \times \{0, 1, \cdots, 400\} \rightarrow \{0, 1, \cdots, 255\}]$ given by

$$\forall (m, n) \in \{0, 1, \cdots, 600\} \times \{0, 1, \cdots, 400\}, \quad x(m, n) = (m + n) \text{ mod } 255$$

(d) $[\{0, 1, \cdots, 600\} \times \{0, 1, \cdots, 400\} \rightarrow \{0, 1, \cdots, 255\}]$ is the appropriate signal space for images of size 600×400 pixels with an 8-bit color map index.

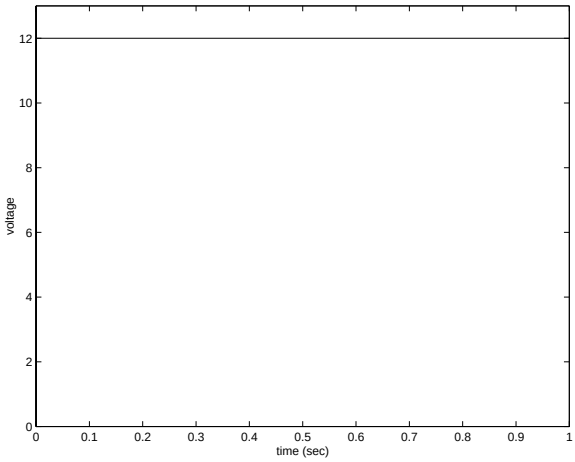
2. (a) One possible model is

$$\textit{CarBatteryVoltage}: \textit{Reals} \rightarrow \textit{Reals}$$

where the domain represents time and the range represents voltage, and

$$\forall\, t \in \textit{Reals}, \quad \textit{CarBatteryVoltage}(t) = 12.$$

Since the value is contant, we could have defined the range to be $\{12\}$, a set with just one element. However, in practice, the voltage is only approximately a constant 12 volts. In fact, while you are starting your car, the electrical load on the battery causes the voltage to drop considerably. The voltage also fluctuates with noise caused by electrical circuits in the car. Thus, setting the range to *Reals* gives us more modeling flexibility, allowing us to be more accurate if we choose to do so. A sketch is shown below:



This is generated by the following Matlab commands

```
plot([0 1],[12 12])
xlabel('time (sec)')
ylabel('voltage')
axis([0 1 0 13])
```

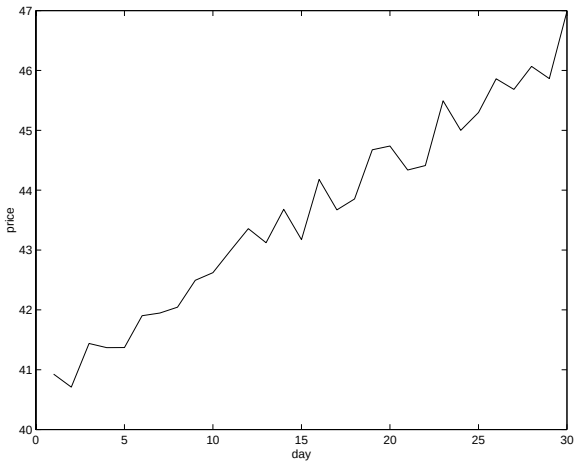
(b) One possible model is

$$price : Naturals \rightarrow Prices$$

where the domain counts off days from some starting point, and the set *Prices* is

$$Prices = \{0, 1/16, 1/8, 3/16, 2/8, 5/16, \cdots\},$$

assuming the stock markets have not yet switched a decimal system. A sketch of a segment of one such possible function (for a fictitious company) is shown below:



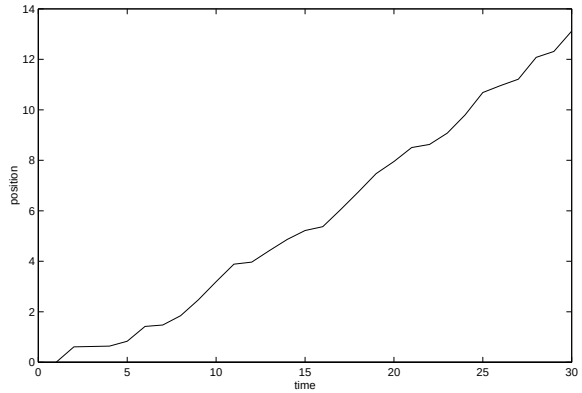
This is generated by the following Matlab commands

```
days = 1:30;
for day = days
   price(day) = 40 + day*0.2 + rand(1);
end
plot(days, price), xlabel('day'), ylabel('price')
```

(c) One possible model is

$$\textit{position}\colon \textit{Reals} \to [0, L]$$

where the domain represents time and L is the length of the road. A sketch of a segment of one such possible function (which assumes that the car is moving forward in fits and starts) is shown below:



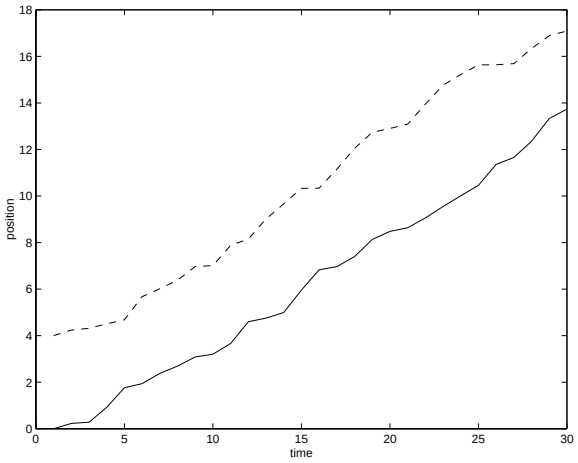
This is generated by the following Matlab commands

```
numSamples = 30;
times = 2:numSamples;
position(1) = 0;
for t = times
   position(t) = position(t-1) + rand(1);
end
plot([1 times], position), xlabel('time'), ylabel('position')
```

(d) One possible model is

$$\textit{twoPositions}\colon \textit{Reals} \to [0, L] \times [0, L]$$

where the domain represents time and L is the length of the road. A sketch of a segment of one such possible function (which assumes that both cars are moving forward in fits and starts) is shown below:



Here, the function is represented as two functions,

$$\textit{onePosition}\colon \textit{Reals} \to [0, L],$$

$$\textit{anotherPosition}\colon \textit{Reals} \to [0, L],$$

such that

$$\forall\, t \in \textit{Reals}, \quad \textit{twoPositions}(t) = (\textit{onePosition}(t), \textit{anotherPosition}(t)).$$

These are both plotted on the same plot, although it would equally valid to plot them in separate plots. This is generated by the following Matlab commands

```
numSamples = 30;
times = 2:numSamples;
position1(1) = 0;
position2(1) = 4;
for t = times
   position1(t) = position1(t-1) + rand(1);
   position2(t) = position2(t-1) + rand(1);
end
plot([1 times], position1, '-', [1 times], position2, '--')
xlabel('time'), ylabel('position')
```

(e) One possible model is

$$\textit{stereoAudio}\colon \textit{Reals} \to \textit{Reals} \times \textit{Reals}$$

where the domain represents time and the range represents air pressure at each of the two ears. Just as with the previous part, we can divide the function into two and plot the two functions separately or on the same plot.

3. The sampling frequency is 44,100 Hz, so the sampling period is $1/44100 \approx 22\ \mu$s.

4. The four signals are shown in figure 1.1.

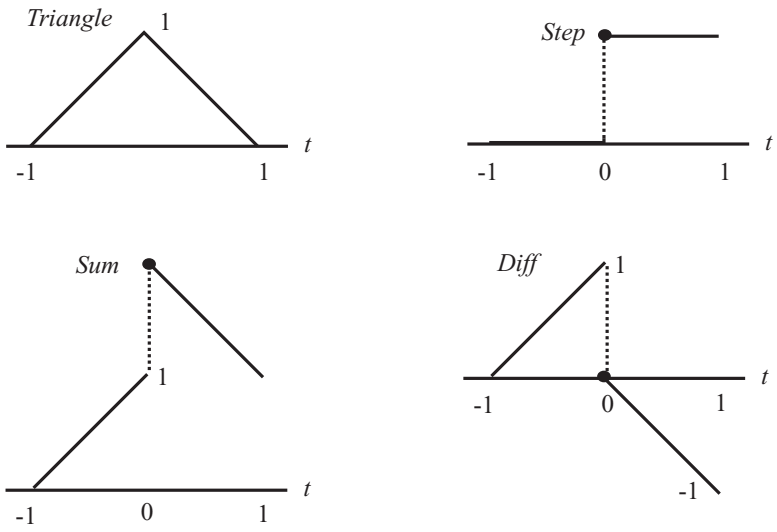


Figure 1.1: The graphs of the signals for Problem 4. Three of the signals are discontinuous at 0; the solid dots marks the value of these signals at 0.

5. (a) One possible model is

$$\textit{PhotoImage}\colon [0, \textit{width}] \times [0, \textit{height}] \to \textit{Intensity}^3$$

where $\textit{Intensity} = [0, \textit{maxIntensity}]$ for some constant *maxIntensity*. The three intensities in the range represent the amount of red, green, and blue, respectively, where 0 represents white (the color is absent) and *maxIntensity* represents the maximum amount of the color that the photographic material can deliver. To get black, all three colors are set at their maximum. Note that an image displayed on a computer screen works in the opposite way. To get black, the three colors are set to their minimum.

(b) The difference here compared to the previous part is that everything is discrete. One possible model is given in the text,

$$\textit{ColorComputerImage}\colon \textit{DiscreteVerticalSpace} \times \textit{DiscreteHorizontalSpace} \to \textit{Integers8}^3$$

so each pixel value is an element of $\{0, 1, \cdots, 255\}^3$.

(c) One possible model is

$$\textit{SurfaceHeight}\colon \textit{Longitudes} \times \textit{Lattitudes} \to [H_{min}, H_{max}]$$

where $\textit{Longitudes} = [0, 360)$ and $\textit{Lattitudes} = [-90, 90]$ (using units of degrees), and $H_{min} = -36,210$ (the altitude in feet of the bottom of the Challenger Gorge of the Mariana Trench), and $H_{max} = 29,028$ (the altitude in feet of the summit of Mount Everest.

(d) One possible model is

$$\textit{ChairLocation}: \{1, 2, \cdots, N\} \rightarrow [0, \textit{roomWidth}] \times [0, \textit{roomLength}],$$

where we have numbered the chairs 1 to N. There are many other valid models. For example, we might construct a function

$$\textit{Chairs}: [0, \textit{roomWidth}] \times [0, \textit{roomLength}] \rightarrow \{0, 1\}$$

and define it so that the value at a location in the room is 0 if there is no chair at that location and 1 if there is. The choice of model depends on how we will use the model. The first model, for example, makes it easier to track chair locations as they move around the room. In the second model, we lose the identity of the chairs, but can easily determine which parts of the room are not covered by some chair.

6. Take the domain to be $[-6, 6] \times [-6, 6]$ for a twelve-inch square with center at $(0, 0)$. Take the range to be *Intensity*^{3} for RGB values. Then for all $(x, y) \in [-6, 6]^2$,

$$\textit{Albers}(x, y) = \begin{cases} (r_y, g_y, b_y), & (x, y) \in [-4, 4]^2 \\ (r_w, g_w, b_w), & \textit{otherwise} \end{cases}$$

where (r_y, g_y, b_y) are the RGB values for the color yellow, and (r_w, g_w, b_w) are the RGB values for white.

7. The number of bits in the image is $1024 \times 768 \times 16 = 12,582,912$. It would take $12582912/28000 = 449$ seconds to download over a 28 Kbps modem, 32.7 seconds over a 384 Kbps modem, and 1.25 seconds over a 10 Mbps LAN.

8. (a) In one possible model, the domain is $\{1, \cdots, 100\}$ and the range is $\{\textit{Heads}, \textit{Tails}\}$. One possible event sequence is

$$\textit{Heads}, \textit{Heads}, \textit{Tails}, \cdots, \textit{Heads}.$$

(b) A good choice for the domain is *Naturals*, since (ideally) the elevator starts functioning at some point in time and never stops. The range might be

$$\{\textit{Basement}, 1, \textit{mezzanine}, 2, 3, 4, 5, \textit{OpenDoor}, \textit{CloseDoor}, \textit{Alarm}\}.$$

Any sequence from this set is a possible event sequence.

(c) Again, a good choice for the domain is *Naturals*, since (ideally) the machine starts functioning at some point in time and never stops. The range might be

$$\{5, 10, 25, \textit{selectCoke}, \textit{select7Up}, \textit{releaseSoda}\}.$$

The numbers represent received coins. Any sequence from this set is a possible event sequence.

(d) A simple choice would be to represent the response as a voice signal. However, this fails to capture the key structure. Directions consist of a sequence of steps. A more detailed model would represent this sequence of steps. A reasonable choice for the domain is *Naturals*, since there is, in theory, no upper bound on the number of steps you might give. The range is a bit more complicated, since we have to decide what are possible directions. Clearly we would want to include things like "go two blocks and turn left on University Avenue." This suggests that the range is a rather large set. It has to include elements "go n blocks" for all $n \in$ *Naturals* and "turn left at a" for all street names a. Obviously we would also need "turn right at a." We may even want to include things like "take a hard left" and "go m miles" for all $m \in$ *Reals*$_+$. If we were constructing, for example, a speech recognition system that could understand spoken directions and then automatically drive your car, we would want to characterize this set very carefully and thoroughly.

(e) A reasonable choice for the domain is *Naturals*, since there is, in theory, no upper bound on the number of moves in a chess game. The range is a fairly large, but finite set. Each of the 64 squares on the board must be identified. Let them be $\{1, \cdots, 64\}$ (not a particularly convenient naming, but adequate for our purposes). Next, each of the 32 game pieces must be identified. Let them be $\{1, \cdots, 32\}$. Then a move can be given as a member of $\{1, \cdots, 64\} \times \{1, \cdots, 32\}$.

9. (a)

$$\textit{Population}: \textit{Cities} \rightarrow \textit{Naturals}$$

where *Cities* is the set of cities.

(b) This problem is a little tricky, since it is possible for there to be two entries in the white pages with the same name, and two entries with the same phone number. Thus, it will not work to map names into phone numbers, nor phone numbers into names. Instead, we choose the domain *Naturals* to represent the position of the white pages entry in the phone book, and the range to be *Names* $\times \{0, 1, \cdots, 9\}^7$.

(c)

$$\textit{Birthdates}: \textit{Students} \rightarrow \textit{Dates}$$

where the sets are obvious.

(d)

$$\textit{Frequencies}: \textit{Stations} \rightarrow [530, 1700]$$

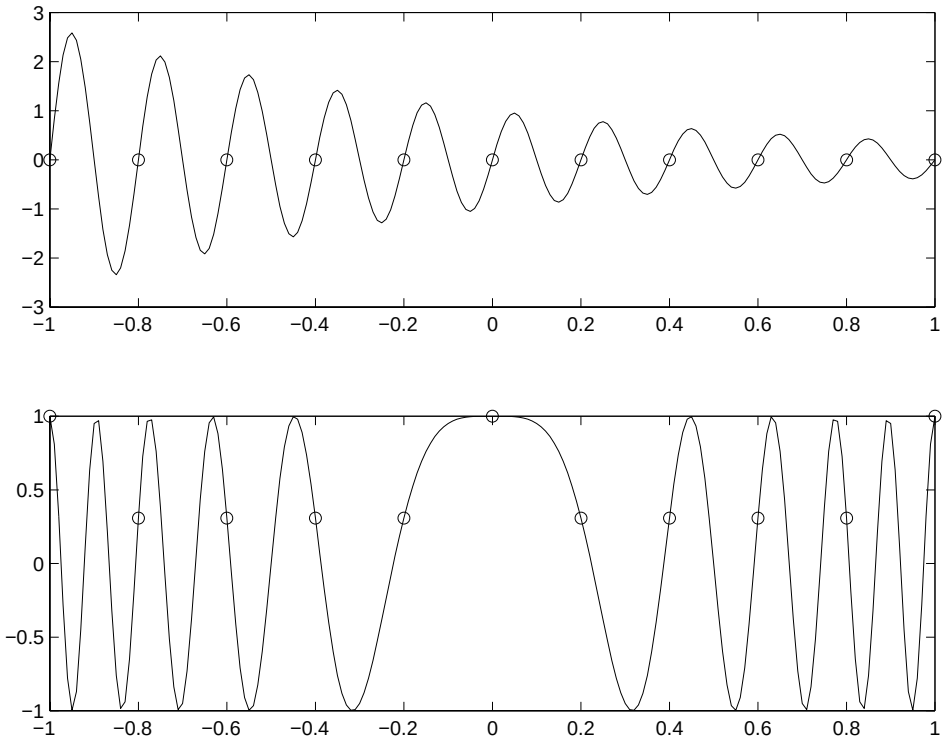
where *Stations* = KDOG, $\cdots$ and the frequencies are in units of kHz. Note that we could actually be more specific and make the range a discrete set, since the FCC only allocates certain discrete frequencies.

(e)

$$\textit{Frequencies}: \textit{Stations} \rightarrow [88, 108]$$

where *Stations* = KQED, KPFA, $\cdots$ and the frequencies are in units of MHz. Note that again we could actually be more specific and make the range a discrete set, since the FCC only allocates certain discrete frequencies.

10. The two plots are shown below:



Notice that in the first case, the samples all have value 0. In the second case, the samples also do not accurately reflect the structure of the waveform. This is a clear indication that we have not sampled frequently enough. The Matlab commands that generated these plots are:

```
denseTime = [-1:0.01:1];
sampleTime = [-1:0.2:1];
y1 = exp(-denseTime).*sin(10*pi*denseTime);
y2 = exp(-sampleTime).*sin(10*pi*sampleTime);
subplot(2,1,1), plot(denseTime,y1,'-',sampleTime,y2,'o');
chirp1 = cos(10*pi*denseTime.^2);
chirp2 = cos(10*pi*sampleTime.^2);
subplot(2,1,2), plot(denseTime,chirp1,'-',sampleTime,chirp2,'o');
```

11. Figure 1.2 gives a plot of the function $\theta : [0, 5] \rightarrow [-\pi, \pi]$. Mathematically, for all $t \in [0, 5]$,

$$\theta(t) = 2\pi t \bmod 2\pi,$$

where for any α, $\alpha \bmod 2\pi = \beta$ is the number $\beta \in [-\pi, \pi)$ such that $\alpha - \beta$ is a multiple of 2π. Although θ is a discontinuous function, the motion is continuous since the position of the pendulum at time t is given by $l \cos \theta(t)$ and $l \sin \theta(t)$ which are continuous in t.

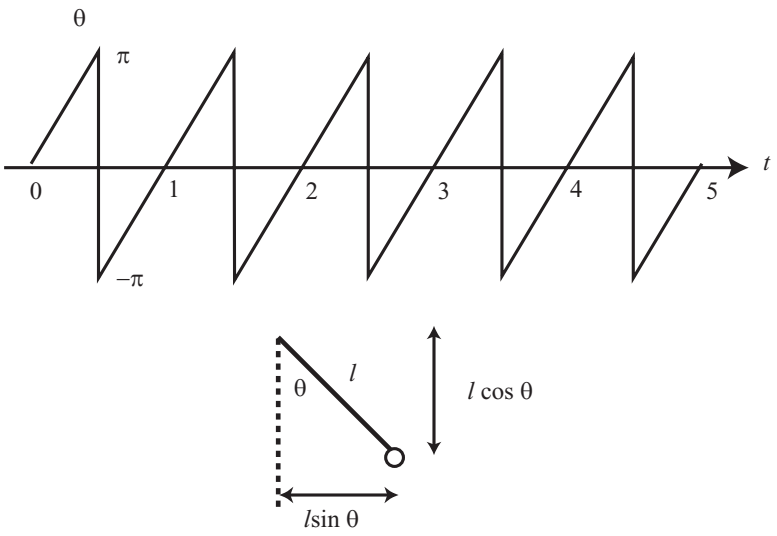


Figure 1.2: Plot of θ for problem 11

12. (a) One way to give an element $f \in [X \to Y]$ is to give a set of pairs $(x, f(x))$, one for each $x \in X$. This way, we can define all the functions in $[X \to Y]$:

$$\{(a,0),(b,0),(c,0)\},$$
$$\{(a,0),(b,0),(c,1)\},$$
$$\{(a,0),(b,1),(c,0)\},$$
$$\{(a,0),(b,1),(c,1)\},$$
$$\{(a,1),(b,0),(c,0)\},$$
$$\{(a,1),(b,0),(c,1)\},$$
$$\{(a,1),(b,1),(c,0)\},$$
$$\{(a,1),(b,1),(c,1)\}.$$

(b) For every element in X, there are n possible values in Y. Since there are m elements in X, the total number of possible functions is n^m. Note that this number grows to be very large for even moderately small finite sets.

(c)

$$256^{6000} = 10^{b \log_{10}(a)} \approx 10^{14449}.$$

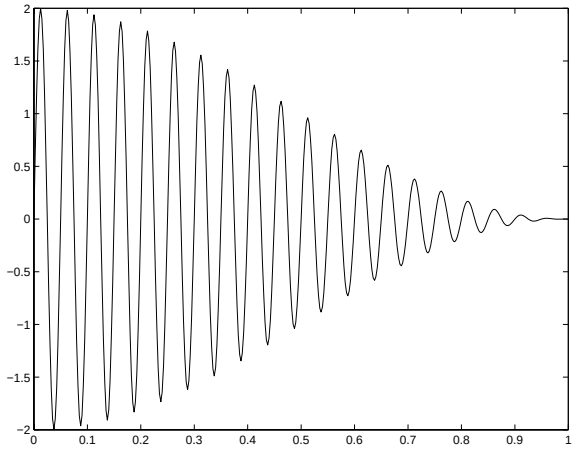
Chapter 2

Defining Signals and Systems – Solutions

1. With the given m and 20 Hz carrier frequency, for all $t \in$ *Time*,

$$\textit{AMSignal}(t) = (1 + \cos(\pi t))sin(2\pi \times 20t).$$

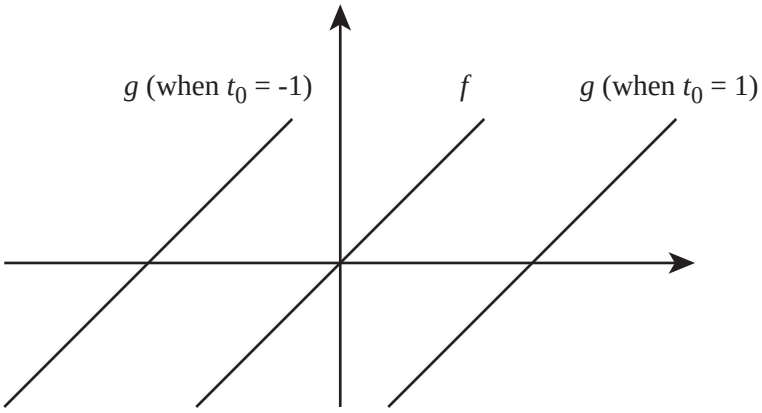
This is plotted for *Time* $= [0, 1]$ below:



The Matlab commands to generate this plot are:

```
t = 0:1/400:1;
AMSignal = (1 + cos(pi * t)).*sin(2*pi*20*t);
plot(t,AMSignal);
```

2. (a) The functions are shown below:



(b) If $(t, y) \in \textit{graph}(f)$ then $f(t) = y$. Hence $g(t + t_0) = f((t + t_0) - t_0) = f(t) = y$, so $(t + t_0, y) \in \textit{graph}(g)$.

(c) If a particular value y occurs for f at time t, then the same value y will occur for g at time $t + t_0$. For $t_0 > 0$ this is later in time, so it is reasonable to call this a delay.

3. (a) false

(b) false

(c) false

(d) false

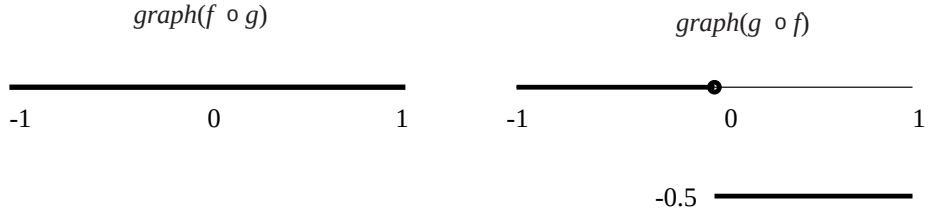
4. For the left graph,

$$\forall\, x \in [-1, 1], \quad f(x) = \begin{cases} 0, & x \in [-1, 0] \\ 2x, & x \in (0, 0.5] \\ 2 - 2x, & x \in (0.5, 1] \end{cases}$$

For the right graph,

$$\forall\, x \in [-1, 1], \quad g(x) = \begin{cases} 1, & x \in [-1, 00 \\ 0, & x = 0 \\ -0.5, & x \in (0, 1] \end{cases}$$

The graphs of the compositions are shown below:



5. (a) Yes, this is the graph of a function. The table is:

$x \in X$	$y \in Y$
a	1
b	1
c	2

(b) This is not the graph of a function because for a in the domain there are two possible values in the range.

(c) This is also not the graph of a function because it is incomplete. It does not give the value of the function for c in the domain.

6. (a) Each entry in the table T is of the form $(d, \textit{outputPort})$, where $d \in D$ and *outputPort* $\in \{O_1, O_2\}$. Thus, the entry $(d, \textit{outputPort}) \in D \times \{O_1, O_2\}$. Hence, $T \subset D \times \{O_1, O_2\}$.

(b) T should be the graph of a function because for each destination address there should be listed exactly one output port, or else the router would not know what to do with the packet.

(c) Let *Packets* $= D \times$ *otherData* be the set of possible packets. The set *otherData* is the set of possible data, other than destination addresses, that the router carries. Define

$$\textit{PacketStream}\colon \textit{Naturals} \to \textit{Packets}.$$

Then an input to the router is in *PacketStream*, and an output is in $\textit{PacketStream}^2$.

(d) The switch is a function

$$\textit{Switch}\colon \textit{PacketStream} \to \textit{PacketStream}^2$$

such that if the input $x \in$ *PacketStream* and $\textit{Switch}(x) = (y, z)$, then y is the substream of x whose destination addresses satisfy $T(d) = O_1$ and z is the substream of x whose destination addresses satisfy $T(d) = O_2$.

7. (a) assignment, assertion, predicate

(b) assignment, assertion, predicate

(c) assertion, predicate

(d) assertion

(e) assertion, predicate

8. Recall from exercise 12 of the previous chapter that if X has m elements and Y has n elements then $[X \to Y]$ has n^m elements. In this case, the size of the domain is 2^m instead of m and the size of the range is 2^n. Thus, the number of possible logic circuits is

$$(2^n)^{(2^m)}.$$

9. (a) The simplest example is the identity system:

$$\forall x, \forall n, \quad H(x)(n) = x(n).$$

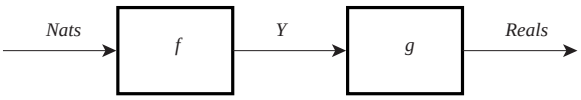
(b) A simple example is a system which converts negative values to 0 and positive values to 1:

$$\forall x, \forall n, \quad H(x)(n) = \begin{cases} 0, & \text{if } x(n) \leq 0 \\ 1, & \text{if } x(n) > 0 \end{cases}$$

(c) A simple example is a "zero-order" hold:

$$\forall x, \forall t \in \textit{Reals}, \quad H(x)(t) = x(n), \text{ where } n = \lfloor t \rfloor.$$

10. (a)



(b)
```
sum(cos(2*pi*[1:100]/x))
```

(c) 100.

11. The following Matlab function can be defined in a file called movingAverage.m:

```
function y = movingAverage(x)
% MOVINGAVERAGE Four-point moving average.
for n=1:3
    y(i) = sum(x(1:n))/4;
end;
N = 4:length(x);
y(N) = (x(N-3) + x(N-2) + x(N-1) + x(N))/4;
```

Notice the way this is written. The for loop calculates the first four samples. The last line calculates all the remaining samples, exploiting the fact that Matlab automatically vectorizes. This yields a very compact definition. An even more compact definition is

```
function y = movingAverage(x)
% MOVINGAVERAGE Four-point moving average.
z = [zeros(1,3), x];
N = 1:length(x);
y(N) = ( z(N) + z(N+1) + z(N+2) + z(N+3))/4;
```

Here, we prepend 3 zeros in front of the argument vector, and then compute the output vector by adding five neighboring values of the expanded vector.

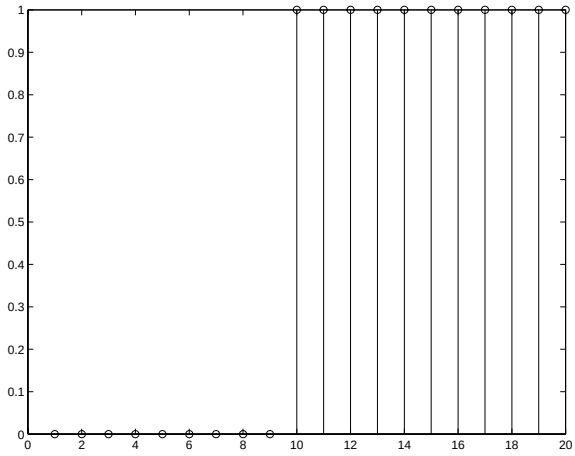
(a) The unit step at 10 can be generated compactly with the commands:

```
N = 1:20;
x = N>=10;
```

The second command assigns to each element of the vector x the result of comparing the corresponding value of t to 10. To check this, plot x with the command:

```
stem(x)
```

which yields:



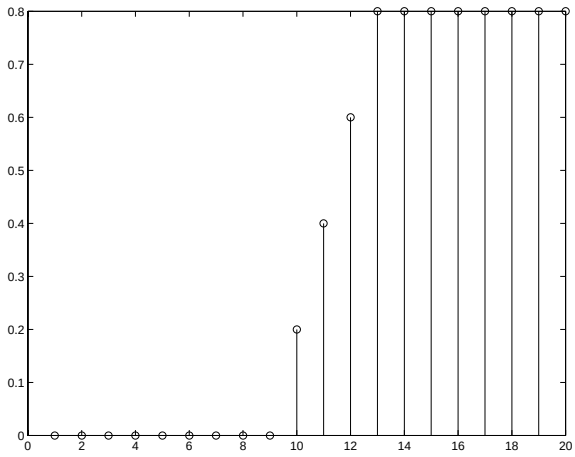
To apply the function we defined above, we need to make sure the m-file storing the function is available to Matlab. The simplest way to do this is to make the current directory in Matlab the same as the directory storing the m-file, e.g., assuming it is directory `d:\users\eal`,

```
>> cd D:\users\eal
>> pwd

ans =

D:\users\eal
```

Then,

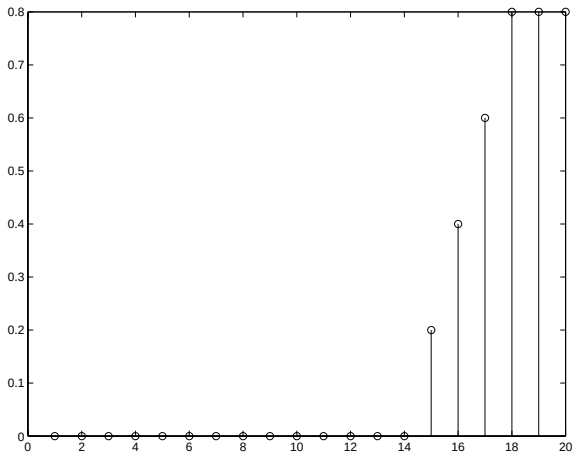
```
stem(movingAverage(x))
```

yields the following plot:



(b)
```
N = 1:20;
x = N>=15;
stem(movingAverage(x))
```

yields the following plot:



Note that is simply delayed by 5 samples, just like the input.

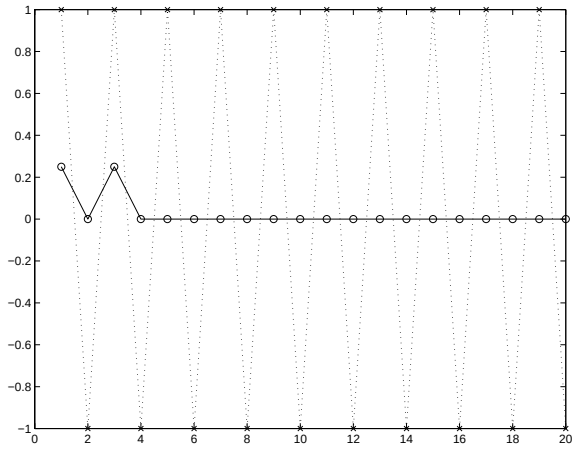
(c) The alternating signal can be generated and the moving average applied by

```
x = cos(pi * [0:19]);
y = movingAverage(x);
```

The result can be plotted together with the original using

```
N = 1:20;
plot(N,x,'x:',N,y,'o-')
```

which yields the result



(Unfortunately, Matlab does not support stem plots with more than one signal, so we use the `plot` command.)

(d) The moving average system changes abrupt transitions into more gradual transitions. In the case of the alternating input, because this signal consists of a repeated sequence of abrupt transitions, the size of the alternation is reduced to zero after the initial transient. The initial transient is due to the fact that the data prior to the first input is assumed to have value zero rather than to be alternating.

12. Using the specified fact from calculus,

$$(F(x))(t) = \frac{1}{10\omega}(\cos(\omega(t-10)) - \cos(\omega t)).$$

From this, we can already see that the amplitude of the output decreases as ω increases because the factor in parenthesis has magnitude no greater than two, and ω is in the denominator of the factor on the left. We can apply the identity in the footnote with

$$\begin{aligned} \alpha &= -10\omega \\ \beta &= 0 \\ A &= 1 \\ B &= -1 \end{aligned}$$

to get

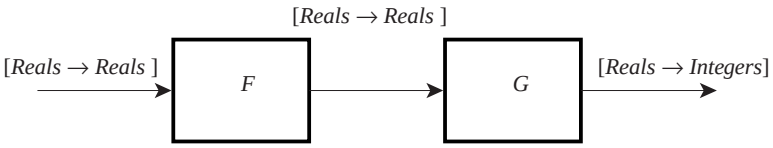
$$(F(x))(t) = \frac{R}{10\omega}\cos(\omega t + \phi)$$

where R and ϕ are (for fixed ω) constants given by the identity. This is sinusoidal with the same frequency ω as the input.

13. (a) h: *Reals* $\rightarrow$ *Integers* such that for all $x \in$ *Reals*,

$$h(x) = \begin{cases} 1 & \text{if } x > -1 \\ 0 & \text{if } x = -1 \\ -1 & \text{if } x < -1 \end{cases}$$

(b) Here is a block diagram:



(c) The desired signal is u: *Reals* $\rightarrow$ *Integers* where for all $x \in$ *Reals*,

$$u(x) = \begin{cases} 0 & \text{if } x \text{ is an odd integer} \\ 1 & \text{otherwise} \end{cases}$$

Note that this signal is 1 almost everywhere and goes to zero only a countable number of discrete points on the real line. No real-world signal would behave this way.

14. $H: D \rightarrow D$ such that for all $x \in D$ and $n \in$ *Integers*, $H(x)$ satisfies

$$(H(x))(n) = x(n) - (H(x))(n-1).$$

This is as much as we can define H. This definition is complete only if such $H(x)$ exists and is unique.

15. (a) Let $y = H(x)$, as in the diagram, and note that $y = 0.5(y + x/y)$. Multiplying through by y we get

$$y^2 = 0.5(y^2 + x)$$

which means that

$$x = y^2$$

so

$$y = \sqrt{x}.$$

(b) The following Matlab sequence shows the result for $x = 4$:

```
>> x = 4;
>> y = 1;
>> y = 0.5*(y + x/y)

y =

    2.5000

>> y = 0.5*(y + x/y)

y =

    2.0500

>> y = 0.5*(y + x/y)

y =

    2.0006

>> y = 0.5*(y + x/y)

y =

    2.0000

>>
```

The following Matlab sequence shows the result for $x = 4$:

```
>> x = 12;
>> y= 1;
>> y = 0.5*(y + x/y)

y =
```

```
    6.5000

>> y = 0.5*(y + x/y)

y =

    4.1731

>> y = 0.5*(y + x/y)

y =

    3.5243

>> y = 0.5*(y + x/y)

y =

    3.4646

>> y = 0.5*(y + x/y)

y =

    3.4641

>> sqrt(12)

ans =

    3.4641

>>
```

Chapter 3

State Machine Models – Solutions

1. (a) This state machine is not deterministic because if it is in state s and input a arrives, then there are two possible transitions to take.

   (b) The *else* arc is a self loop (both the source and destination states are s), and $\mathit{else} = \{c, \mathit{absent}\}$.

2. (a) The state response is

$$(\mathit{idle}, \mathit{idle}, \mathit{idle}, \mathit{count1}, \mathit{idle}, \mathit{count1}, \mathit{count2}, \mathit{play\ greeting}, \mathit{idle}, \cdots).$$

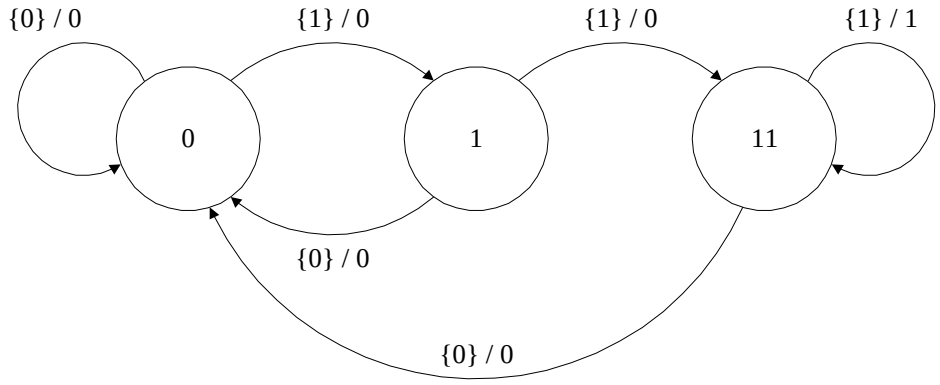
   (b) The trace is

$$\begin{aligned} &\mathit{idle} \xrightarrow{\mathit{offhook}} \mathit{idle} \xrightarrow{\mathit{offhook}} \mathit{idle} \xrightarrow{\mathit{ring}} \mathit{count1} \xrightarrow{\mathit{offhook}} \mathit{idle} \\ &\xrightarrow{\mathit{ring}} \mathit{count1} \xrightarrow{\mathit{ring}} \mathit{count2} \xrightarrow{\mathit{ring}} \mathit{play\ greeting} \xrightarrow{\mathit{offhook}} \mathit{idle} \cdots \end{aligned}$$

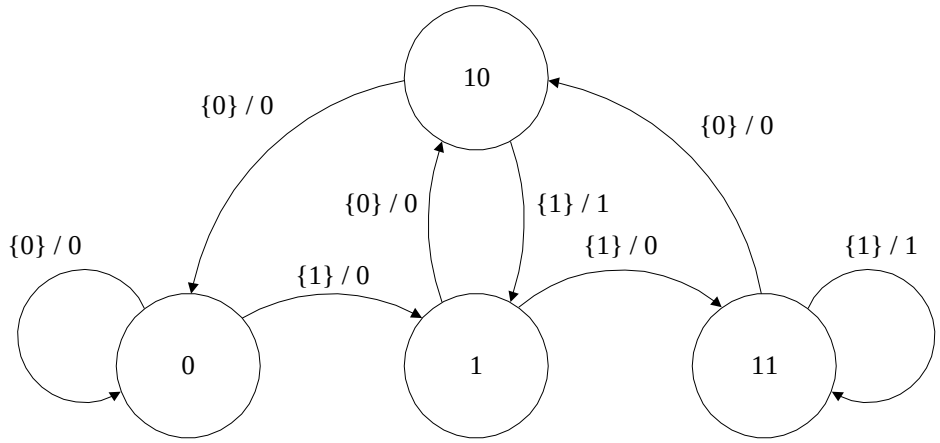
   (c) The output is

$$(\mathit{absent}, \mathit{absent}, \mathit{absent}, \mathit{absent}, \mathit{absent}, \mathit{absent}, \mathit{answer}, \mathit{absent}, \cdots).$$

3. The solution is shown below:



(a)



(b)

For part (a), *States* $= \{0, 1, 11\}$ while for part (b), *States* $= \{0, 1, 11, 10\}$.

4. (a) The state transition diagram for the mod-4 counter is shown in figure3.1.

   (b) The update table is:

state	*next state/output under input*			
	increment	*decrement*	*reset*	*absent*
0	1/1	$3/3$	0/0	0/*absent*
1	2/2	0/0	0/0	1/*absent*
2	3/3	1/1	0/0	2/*absent*
3	0/0	2/2	0/0	3/*absent*

   (c) The set of states is

$$States = \{0, 1, 2, 3\}.$$

   The set of inputs is

$$Inputs = \{increment, decrement, reset, absent\}.$$

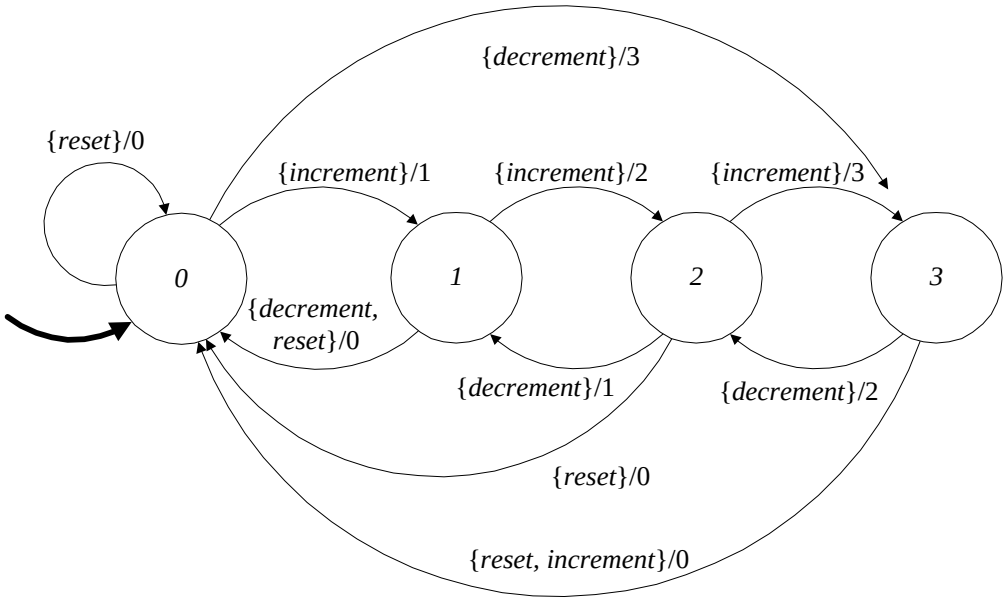


Figure 3.1: Solution to exercise 4.

The set of outputs is

$$\mathit{Outputs} = \{0, 1, 2, 3, \mathit{absent}\}.$$

The update function

$$\mathit{update}\colon \mathit{States} \times \mathit{Inputs} \to \mathit{States} \times \mathit{Outputs}$$

is given by the table in part (b).

(d) The state response for the input sequence $\mathit{increment}^4, \mathit{decrement}^3, \ldots$ is

$$1, 2, 0, 1, 2, 1, 0, 2, \cdots.$$

5. (a) The state machine is shown in figure 3.2. The stuttering transitions are all implicit. The definition is:

$$\begin{aligned} \mathit{States} &= \{a, b, c\} \\ \mathit{Inputs} &= \{a, b, c, \mathit{absent}\} \\ \mathit{Outputs} &= \{a, b, c, \mathit{absent}\} \\ \mathit{initialState} &= a \end{aligned}$$

and $\forall\ (s, x) \in \mathit{States} \times \mathit{Inputs}$,

$$\mathit{update}(s, x) = (x, s).$$

(b) The past history that needs to be summarized is only the previous input. But since the input set is infinite, then the number of patterns needed to summarize that past history is

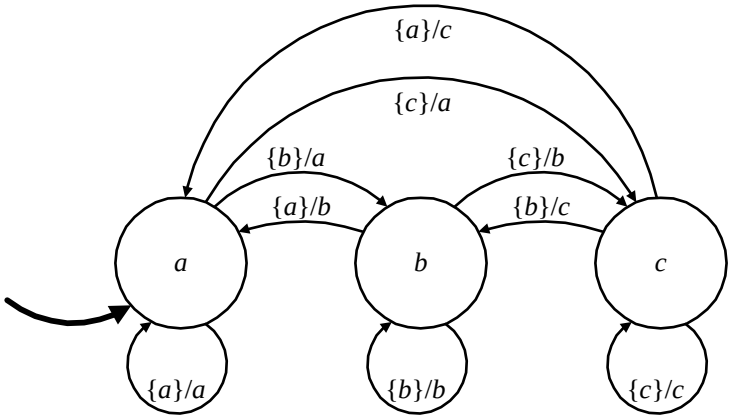


Figure 3.2: Solution to exercise 5(a).

infinite. An infinite state machine for this instance of *UnitDelay* is similar to the finite state machine in part (a), with the only differences being

$$\begin{aligned} \textit{States} &= \textit{Naturals} \\ \textit{Inputs} &= \{absent\} \cup \textit{Naturals}_0 \\ \textit{Outputs} &= \{absent\} \cup \textit{Naturals}_0 \end{aligned}$$

6. From the description of *Equal*, it is clear that

$$\textit{Inputs} = \{0, 1, \textit{absent}\},$$

and

$$\textit{Outputs} = \{\textit{equal}, \textit{notEqual}, \textit{absent}\}.$$

We can define the state to be difference between the number of ones that have arrived and the number of zeros that have arrived, in which case,

$$\textit{States} = \textit{Integers}.$$

Then for all $s \in \textit{States}$ and $x \in \textit{Inputs}$,

$$\textit{update}(s, x) = \begin{cases} (0, \textit{equal}) & \text{if } (x = 1 \wedge s = -1) \vee (x = 0 \wedge s = 1) \\ (s + 1, \textit{notEqual}) & \text{if } x = 1 \wedge s \neq -1 \\ (s - 1, \textit{notEqual}) & \text{if } x = 0 \wedge s \neq 1 \\ (s, \textit{absent}) & \text{otherwise} \end{cases}$$

7. When designing a FSM, first consider the general operation and possible complications for the machine, and define the possible inputs and outputs. Then, create the necessary states and state transitions. You also need to choose the level of detail at which you do the modeling. For this problem, for example, is there any point in modeling the position of the elevator between floors? Probably not. The solution given here is certainly not unique. There are many other equally valid designs.

A controller for an elevator should manage the elevator's basic movements. An elevator has three fundamental motions: move up (*moveUp*), move down (*moveDown*), and stop (*stop*). These three actions, plus the stuttering element, constitute the outputs for the FSM. Hence

$$Outputs = \{stop, moveUp, moveDown, absent\}.$$

Two of the inputs to the FSM are well defined as requests for service on floor one and floor two. These two inputs instruct the elevator to begin moving. Input is also needed to instruct the elevator to stop moving when it has arrived at floor. It is acceptable to assume that detectors exist at each floor to inform the FSM of the elevator's position. With the stuttering element, the four possible inputs are

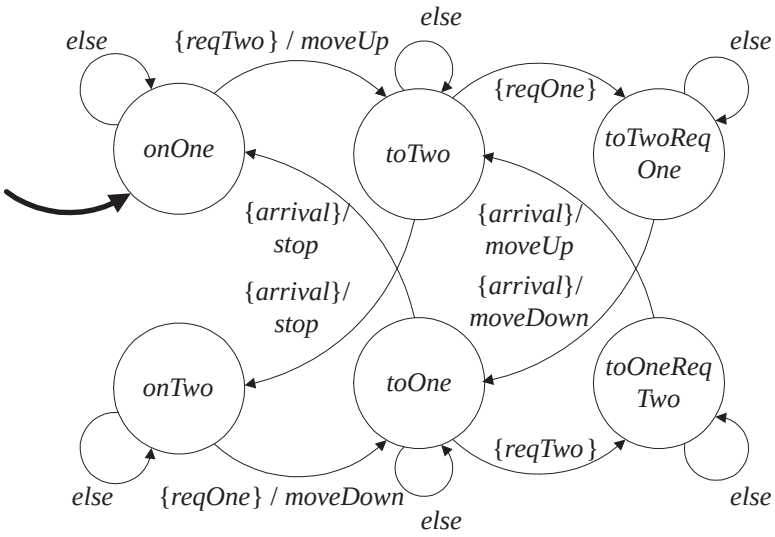
$$Inputs = \{reqOne, reqTwo, arrival, absent\},$$

which are a request for service to floor one, a request for service to floor two, and notification of arrival at a floor.

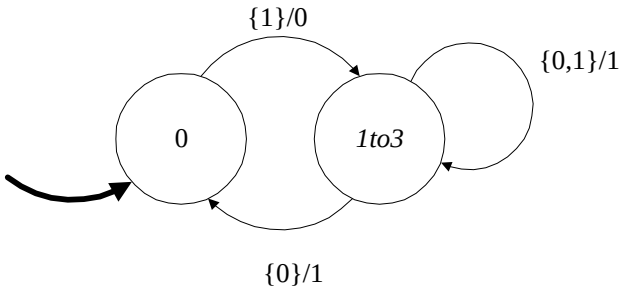
Because all requests should be served, special cases exist if requests for service are received while the elevator is in motion. If a request for service is received for the destination floor, the request is redundant and can be safely ignored. If a request for service is received for the other floor, the request should be stored for later service. Subsequent requests for that floor are redundant and can be ignored, as well. So, for each floor there are three basic states: stopped on the floor (*onOne*, *onTwo*), moving to the floor (*toOne*, *toTwo*), and moving to the floor with a request for the other floor (*toOneReqTwo*, *toTwoReqOne*). Thus,

$$States = \{onOne, onTwo, toOne, toTwo, toOneReqTwo, toTwoReqOne\}.$$

With the operation of the elevator understood and the states, inputs, and outputs well defined, the definition of the state transition function is clear. For this FSM, a graph is the most intuitive representation for the state transitions:



8. The following machine does the job:



As suggested by the state names, *1to3* matches states 1, 2, and 3, while 0 matches 0.

9. (a) The *possibleUpdates* function is given by the following table:

state	*next state/output under input*		
	0	1	*absent*
A	$\{(A, 0)\}$	$\{(B, 1), (C, 0)\}$	$\{(A, \textit{absent})\}$
B	$\{(B, 1)\}$	$\{(B, 1)\}$	$\{(B, \textit{absent})\}$
C	$\{(C, 0)\}$	$\{(C, 0)\}$	$\{(C, \textit{absent})\}$

(b) One way to define this relation is as follows (ignoring suttering). Define the input signals

$$\textit{zeros} = (0, 0, \cdots)$$

and

$$x_M = (\underbrace{0, \cdots, 0}_{M}, 1, \text{anything}, \cdots)$$

where $M \in \textit{Naturals}_0$. That is, x_M is a sequence that starts with zero or more 0's (M of them), followed by a 1, followed by any sequence of 0's and 1's. Note that since we are ignoring stuttering, all possible inputs have this form. Similarly, define

$$y_M = (\underbrace{0, \cdots, 0}_{M}, 1, 1, \cdots)$$

which is a sequence starting with zero or more 0's, followed by all 1's. Now we can define the relation

$$\textit{Behaviors} = \{(x_M, y) \mid (M \in \textit{Naturals}_0) \wedge (y = \textit{zeros} \vee y = y_M)\} \cup \{(\textit{zeros}, \textit{zeros})\}.$$

10. The bisimulation relation is

$$\{(\textit{start}, \textit{start}), (1, 1), (11, 11), (110, 110), (\textit{start}, 1100)\}.$$

11. (a) no
(b) yes

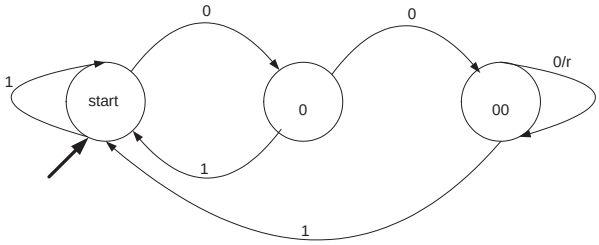


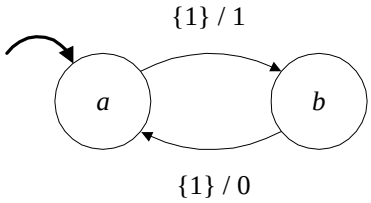
Figure 3.3: Machine which recognizes 000

(c) no

(d) yes

(e) no

12. A two-state bisimilar machine is shown below:



The bisimulation relation is

$$S = \{(a, a), (b, b), (c, a), (d, b)\},$$

or equivalently,

$$S' = \{(a, a), (b, b), (a, c), (b, d)\},$$

13. Yes, we can give such a machine B. It has one state; let's call it e, with two self loops labeled:

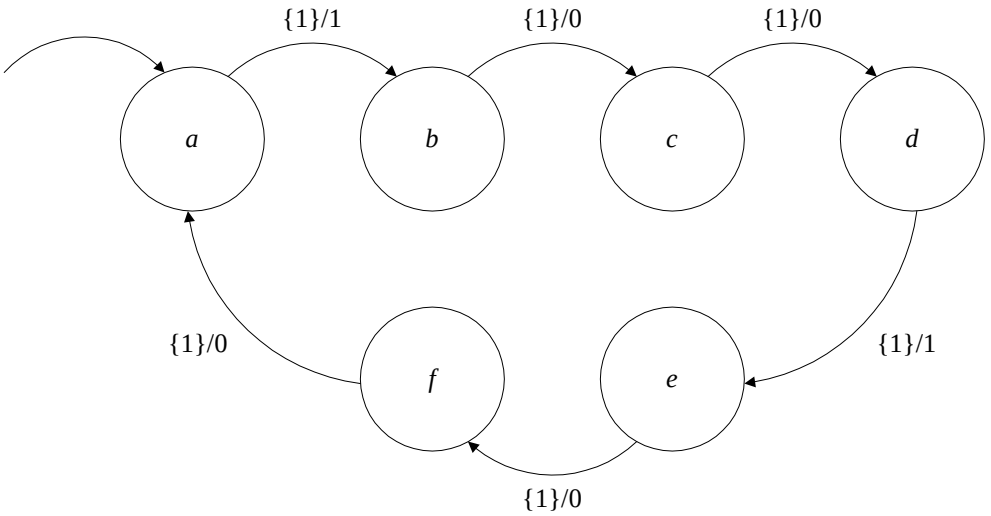
$$\textit{Inputs}/1$$

$$\textit{Inputs}/2$$

The simulation relation is

$$S = \{(a, e), (b, e), (c, e), (d, e)\}.$$

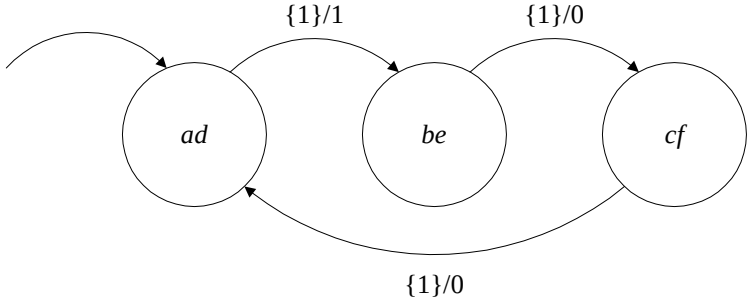
14. We need to remember the patterns 1, 0, 00. The machine is given in figure 3.3.



15. (a)

(b)

$$\textit{Behaviors} = \{(x, y) \mid x = (1, 1, 1, \cdots) \ \wedge \ y = (1, 0, 0, 1, 0, 0, 1, 0, 0, \cdots)\}.$$



(c)

The bisimulation relation is

$$\{(a, ad), (b, be), (c, cf), (d, ad), (e, be), (f, cf)\}$$

Chapter 4

Composing State Machines

1. **Assumptions about the component machines:**
   $\mathit{Outputs}_{A2} \subset \mathit{Inputs}_{B1}$

   Definition of the composite machine:
   $\mathit{States} = \mathit{States}_A \times \mathit{States}_B$
   $\mathit{Inputs} = \mathit{Inputs}_A \times \mathit{Inputs}_{B2}$
   $\mathit{Outputs} = \mathit{Outputs}_{A1} \times \mathit{Outputs}_{A2} \times \mathit{Outputs}_B$
   $\mathit{initialState} = (\mathit{initialState}_A, \mathit{initialState}_B)$
   $\mathit{update}((s_A, s_B), (x_A, x_{B2})) = ((s'_A, s'_B), (y_{A1}, y_{A2}, y_B))$
   where $(s'_A, (y_{A1}, y_{A2})) = \mathit{update}_A(s_A, x_A)$
   and $(s'_B, y_B) = \mathit{update}_B(s_B, (y_{A2}, x_{B2}))$

2. (a) **Assumptions about the component machines:**
   $\mathit{Outputs}_A \subset \mathit{Inputs}_B \subset \mathit{Inputs}_C$
   Definition of the composite machine:
   $\mathit{States} = \mathit{States}_A \times \mathit{States}_B \times \mathit{States}_C$
   $\mathit{Inputs} = \mathit{Inputs}_A$
   $\mathit{Outputs} = \mathit{Outputs}_C$
   $\mathit{initialState} = (\mathit{initialState}_A, \mathit{initialState}_B, \mathit{initialState}_C)$
   $\mathit{update}((s_A, s_B, s_C), x) = ((s'_A, s'_B, s'_C), y_C)$
   where
   $(s'_A, y_A) = \mathit{update}_A(s_A, x)$, $(s'_B, y_B) = \mathit{update}_B(s_B, y_A)$, and
   $(s'_C, y_C) = \mathit{update}_C(s_C, y_B)$.

   (b) For the first composition (A and B):
   Assumptions about the component machines:
   $\mathit{Outputs}_A \subset \mathit{Inputs}_B$
   Definition of the composite machine:
   $\mathit{States}_D = \mathit{States}_A \times \mathit{States}_B$
   $\mathit{Inputs}_D = \mathit{Inputs}_A$

$\mathit{Outputs}_D = \mathit{Outputs}_B$
$\mathit{initialState}_D = (\mathit{initialState}_A, \mathit{initialState}_B)$
$\mathit{update}_D((s_A, s_B), x) = ((s'_A, s'_B), y_B)$
where

$(s'_A, y_A) = \mathit{update}_A(s_A, x)$ and
$(s'_B, y_B) = \mathit{update}_B(s_B, y_A)$.

For the second composition (D and C):
Assumptions about the component machines:
$\mathit{Outputs}_D \subset \mathit{Inputs}_C$
Definition of the composite machine:
$\mathit{States} = \mathit{States}_D \times \mathit{States}_C$
$\mathit{Inputs} = \mathit{Inputs}_D$
$\mathit{Outputs} = \mathit{Outputs}_C$
$\mathit{initialState} = (\mathit{initialState}_D, \mathit{initialState}_C)$
$\mathit{update}((s_D, s_C), x) = ((s'_D, s'_C), y_C)$
where

$(s'_D, y_D) = \mathit{update}_D(s_D, x)$ and $(s'_C, y_C) = \mathit{update}_C(s_C, y_D)$.

(c) The models in (a) and (b) are different but equivalent state machines.

3. (a) Assuming the solution for *UnitDelay*,

$$\begin{aligned} \mathit{States} &= \{a, b, c\} \times \{a, b, c\} \\ \mathit{Inputs} &= \{a, b, c, \mathit{absent}\} \\ \mathit{Outputs} &= \{a, b, c, \mathit{absent}\} \\ \mathit{initialState} &= (a, a) \end{aligned}$$

and $\forall\ (s_1, s_2) \in \mathit{States},\ x \in \mathit{Inputs}$,

$$\mathit{update}((s_1, s_2), x) = \begin{cases} ((s_1, s_2), \mathit{absent}) & \text{if } x = \mathit{absent} \\ ((x, s_1), s_2) & \text{otherwise} \end{cases}$$

(b) Yes, all states are reachable.

(c) For three machines:

$$\mathit{States} = \{a_1, b_1, c_1\} \times \{a_2, b_2, c_2\} \times \{a_3, b_3, c_3\}.$$

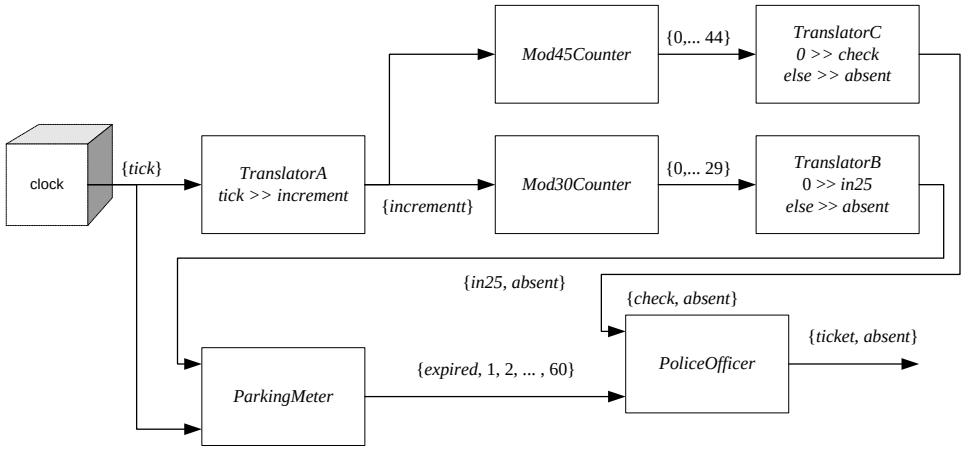
To determine how many elements there are in this set, consider first the product of the first two. For each element of the set $\{a_1, b_1, c_1\}$ there are three elements of the set $\{a_2, b_2, c_2\}$. Thus, this first product has $3 \times 3 = 9$ elements. For each of the 9 elements of this set, there are three elements of the set $\{a_3, b_3, c_3\}$, so the set *States* has $3 \times 3 \times 3 = 27$ elements.
For four machines:

$$\begin{aligned} \mathit{States} &= \{a_1, b_1, c_1\} \times \{a_2, b_2, c_2\} \\ &\quad \times \{a_3, b_3, c_3\} \times \{a_4, b_4, c_4\}. \end{aligned}$$

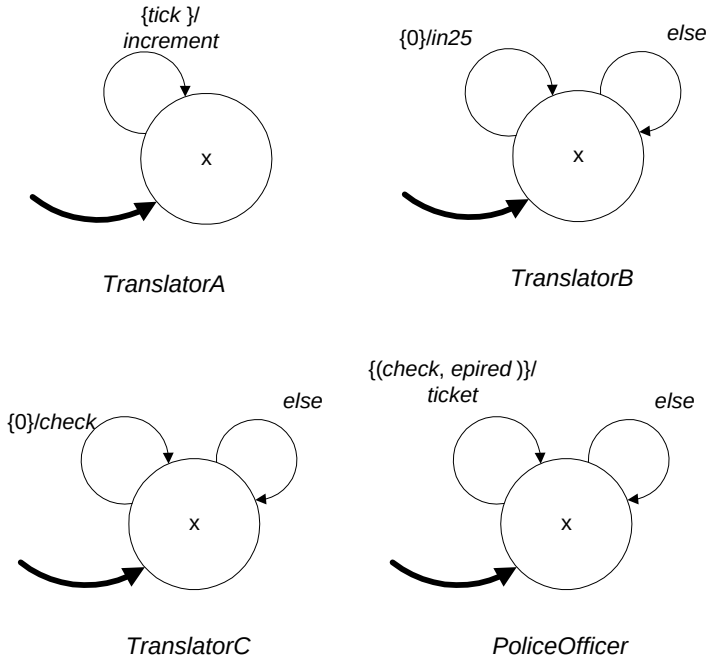
Following a similar argument to that above, this set has $3^4 = 81$ elements.

(d) The pattern in part (c) suggests the answer. For a cascade of n delays, there are 3^n states.

4. The block diagram is shown below:



The state machines are:

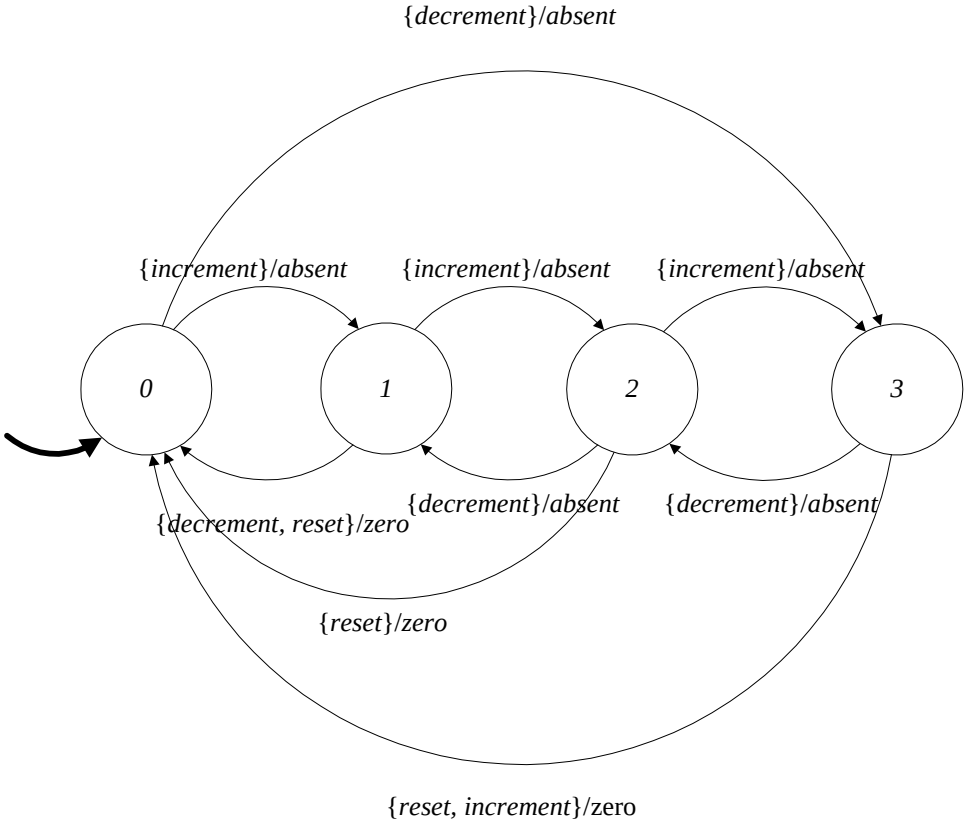


The *ParkingMeter* machine is the same as the solution in the previous chapter, except that the input alphabet is factored. The two modulo N counters are identical to those in the previous chapter, except for the value of N.

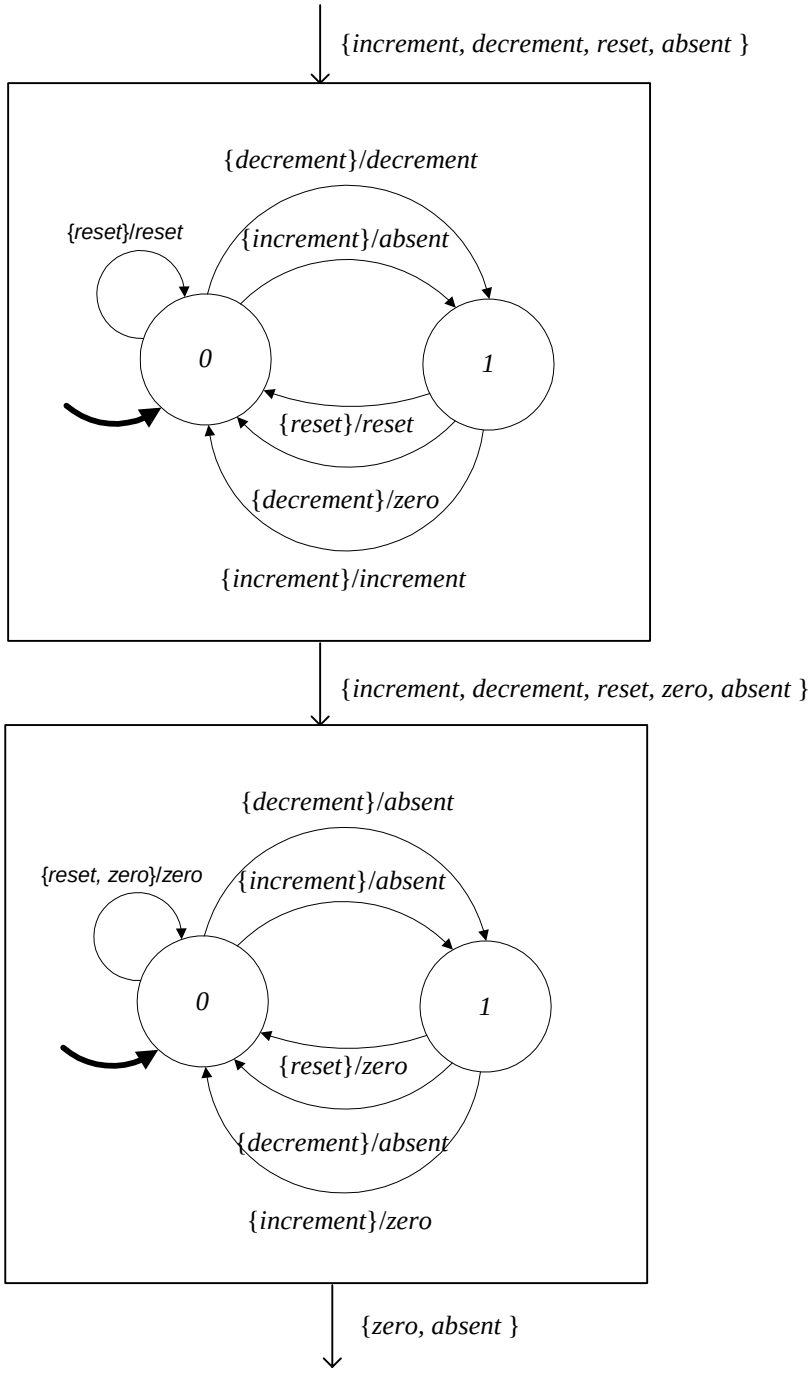
5. (a) The update function is

$$\mathit{update}(s, x) = \begin{cases} (0, \mathit{zero}) & \text{if } s = 1 \text{ and } x = \mathit{decrement} \\ (3, \mathit{absent}) & \text{if } s = 0 \text{ and } x = \mathit{decrement} \\ (s - 1, \mathit{absent}) & \text{if } s > 1 \text{ and } x = \mathit{decrement} \\ (0, \mathit{zero}) & \text{if } s = 3 \text{ and } x = \mathit{increment} \\ (s + 1, \mathit{absent}) & \text{if } s \neq 3 \text{ and } x = \mathit{increment} \\ (0, \mathit{zero}) & \text{if } x = \mathit{reset} \\ (s, \mathit{absent}) & \text{if } x = \mathit{absent} \end{cases}$$

A sketch is shown below:



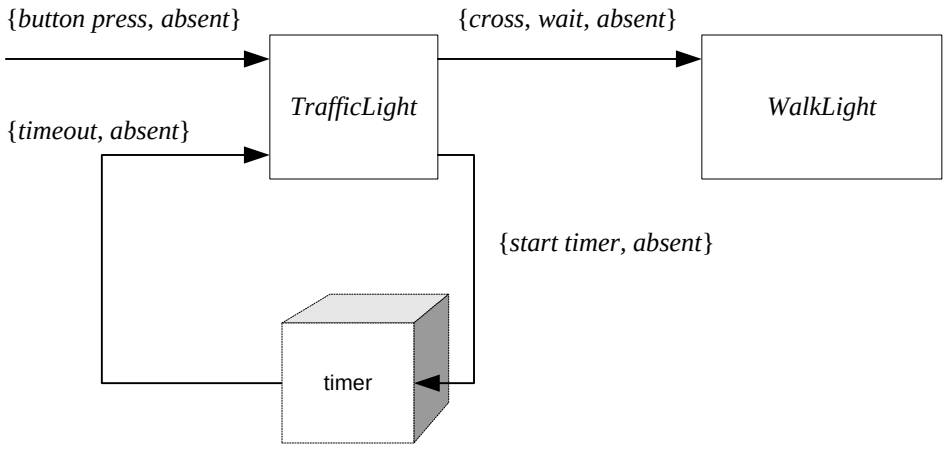
(b) A sketch is shown below:



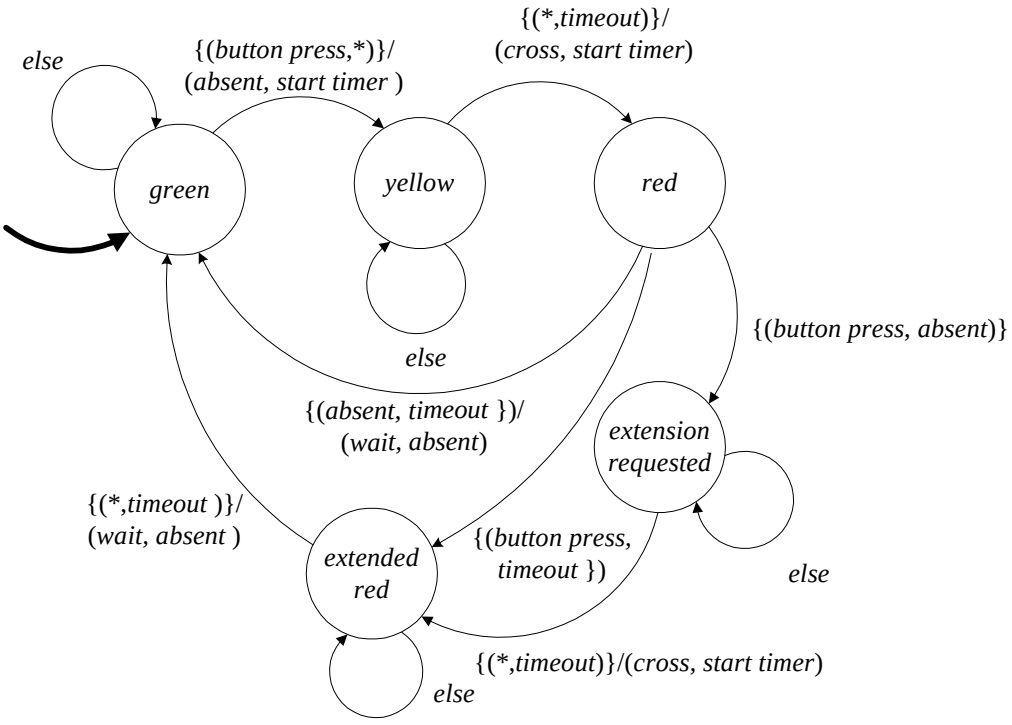
(c) The bisimulation relation is

$$\{(0,(0,0)),(1,(1,0)),(2,(0,1)),(3,(1,1))\}.$$

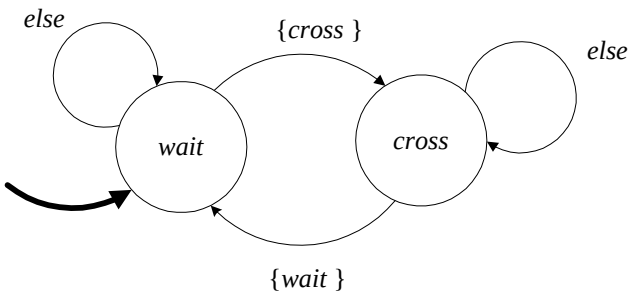
6. One possible design structures the state machines as follows:



where the state transition diagram for the traffic light model is:



and the state transition diagram for the wait light is:



This assumes that the crossing light starts in the "wait" state and that outputs from this state machine change it.

The set of states for the traffic light is

$$\textit{States} = \{\textit{green}, \textit{yellow}, \textit{red}, \textit{extension requested}, \textit{extended red}\},$$

The set of inputs is

$$\textit{Inputs} = \{\textit{button press}, \textit{absent}\} \times \{\textit{timeout}, \textit{absent}\}.$$

The set of outputs is

$$\textit{Outputs} = \{\textit{start timer}, \textit{absent}\} \times \{\textit{wait}, \textit{cross}, \textit{absent}\}.$$

The *WaitLight* machine is more trivial:

$$\textit{States} = \{\textit{wait}, \textit{cross}\},$$

$$\textit{Inputs} = \{\textit{wait}, \textit{cross}, \textit{absent}\}.$$

There are no outputs, so

$$\textit{Outputs} = \{\textit{absent}\}.$$

7. In each case, let i denote the size of the input alphabet of the composition, o the size of the output alphabet, and s the number of states.

   (a) $i = i_A i_B$, $o = o_A o_B$, and $s = s_A s_B$.

   (b) $i = i_A$, $o = o_B$, and $s = s_A s_B$.

   (c) $i = 2$, $o = o_A$, and $s = s_A s_B$.

8. This algorithm and its variants are used to verify properties such as "all states are reachable."

   (a) By definition, *ReachableStates*$(n+1)$ is the union of some set and *ReachableStates*(n), so that certainly *ReachableStates*(n) is a subset of *ReachableStates*$(n+1)$.

   (b) We first show by induction that *ReachableStates*(n) is the set of states reachable from *initalState* in n or fewer steps. This is certainly true for $n = 0$, since by definition, *ReachableStates*$(0) = \{$*initialState*$\}$. Suppose the assertion is true for some $n \geq 0$, and consider the assertion for $n + 1$.

Suppose $s(n+1) \in$ *ReachableStates*$(n+1)$. By definition, either $s(n+1) \in$ *ReachableStates*(n) in which case $s(n+1)$ is reachable from *initialState* in n or fewer steps, or there exists a state $s(n) \in$ *ReachableStates*(n) from which $s(n+1)$ can be reached in one step, so that $s(n+1)$ can be reached from *initialState* in $n+1$ or fewer steps.

On the other hand if $s(n+1)$ is reachable from *initialState* in $n+1$ or fewer steps, then there is a sequence of state transitions

$$\textit{initialState} \to s(1) \to s(2) \cdots \to s(k) \to s(n+1),$$

with $k \leq n$, in which case $s(k) \in$ *ReachableStates*(n) so that, by definition, $s(n+1) \in$ *ReachableStates*$(n+1)$. The assertion now follows by induction.

Now suppose

$$\textit{ReachableStates}(n) = \textit{ReachableStates}(n+1). \tag{4.1}$$

We show that

$$\textit{ReachableStates}(n+1) = \textit{ReachableStates}(n+2).$$

Suppose $s(n+1) \in$ *ReachableStates*$(n+2)$. Then either $s(n+1)$ is already in *ReachableStates*$(n+1)$, in which case we are done, or there is a state $s(n) \in$ *ReachableStates*$(n+1)$ from which $s(n+1)$ can be reached in one step. But, by hypothesis, $s(n)$ is reachable in n steps. It follows that $s(n+1)$ is reachable in $n+1$ steps, i.e. it is in *ReachableStates*$(n+1)$. We can repeat the argument and show that *ReachableStates*$(n) = \cdots =$ *ReachableStates*$(n+k)$.

(c) If (4.1) holds for $n = N$, then

$$\textit{ReachableStates}(N) = \textit{ReachableStates}(N+k),$$

for all k. So if a state $s(n)$ is reachable in any number of steps, it is reachable in at most N steps, hence *ReachableStates*(N) is the set of all reachable states.

(d) Suppose a state $s(n)$ is reachable in $N+k$ steps but not in fewer steps for some $k \geq 0$. So there is a sequence of transitions,

$$\textit{initialState} = s(0) \to s(1) \to \cdots \to s(N+k) = s(n).$$

But since there are N states in all, it must be the case that $s(i) = s(j)$ for some $0 \leq i < j \leq N+k$. Then there is also a sequence of transitions

$$s(0) \to \cdots \to s(i) \to s(j+1) \to \cdots \to s(N+k),$$

so that $s(n)$ is reachable in $N+k-(j-i) < N+k$ steps which is a contradiction.

(e) For the example we have

$$\textit{ReachableStates}(0) = \{(0,0)\}, \text{ and } \textit{ReachableStates}(n) = \{(0,0),(1,1)\}, n \geq 1.$$

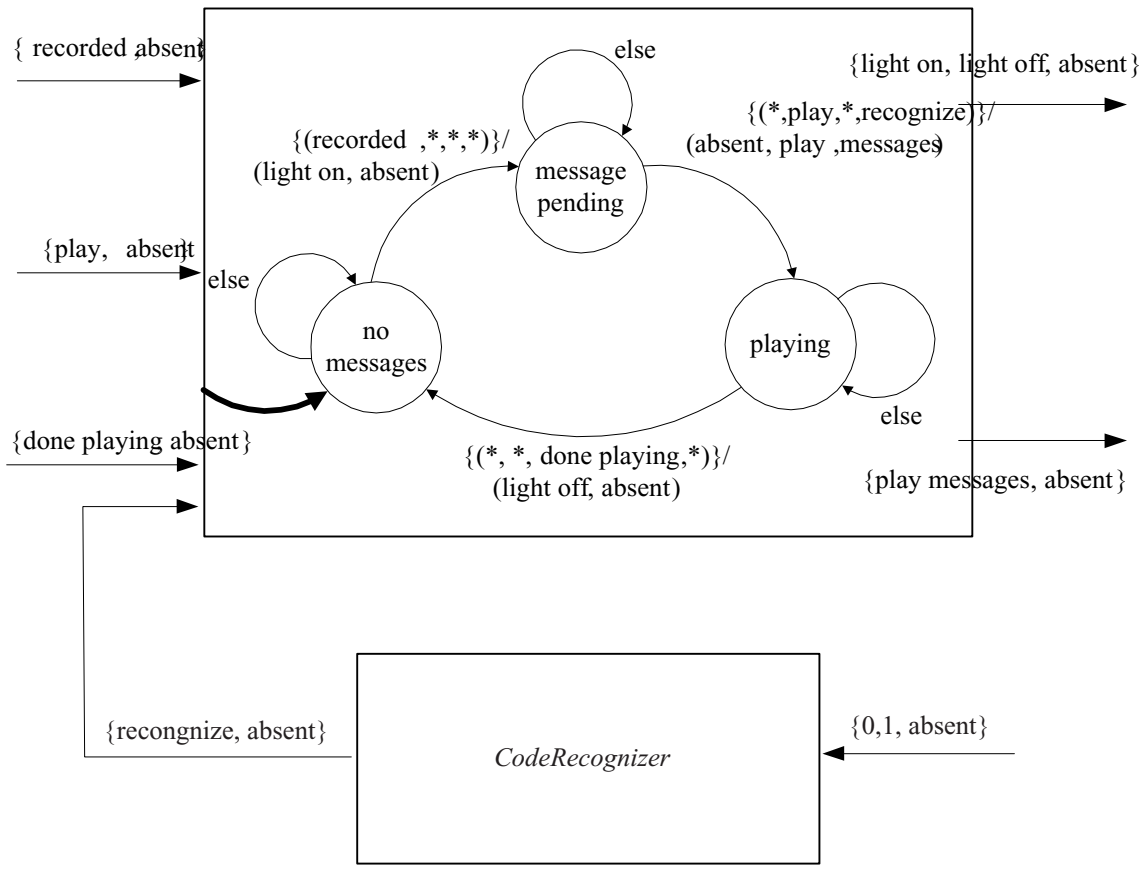
(f) We have seen that *ReachableStates*(n) is the set of all states reachable in n or fewer steps. The result follows.

9. (a) From the definition of *nextStep* we have *nextStep*$(\emptyset) = \emptyset$, and *nextStep*$(States) = States$, so $\emptyset$ and *States* are indeed fixed points.

   (b) By definition, *ReachableStates* $\subset$ *nextState*$(ReachableStates)$. On the other hand if $(s(n+1), y(n)) \in$ *possibleUpdates*$(s(n), x(n))$ and $s(n)$ is reachable, then $s(n+1)$ is also reachable, so that *nextState*$(ReachableStates) \subset ReachableStates$. So *ReachableStates* is a fixed point.

   (c) Let S be a fixed point with *initialState* $\in S$. Then *nextState*$(initialState) \subset$ *nextState*$(S) = S$, so that a state that is reachable in one step is in S. Repeating this, we see that a state that is reachable in any number of steps is in S, and hence *ReachableStates* $\subset S$. So *ReachableStates* is indeed the smallest (least) fixed point of *nextStep* that contains *initialState*.

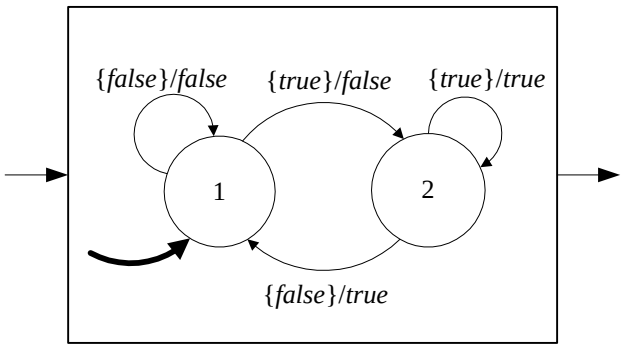
10. The composite state machine is



   Observe that the playback machine is modified to include an additional input port connected to *CodeRecognizer*.

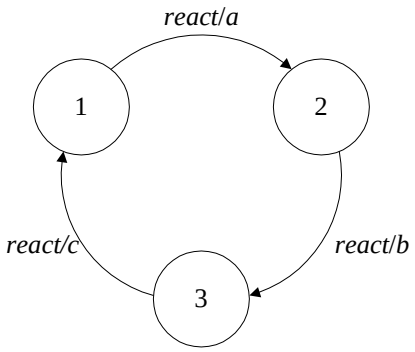
11. Yes, it is well formed because in each state, even if the input is unknown, the output can be determined. The output sequence for the first 10 reactions is $(2, 3, 1, 2, 3, 1, 2, 3, 1, 2)$.

12. (a) The state machine for the delay is



(b) In both states of the delay machine of part (a), the next output is known. Thus, the output can always be determined even if the input is unknown. Thus the procedure for finding values converges in at most two rounds.

13. A solution is shown below:



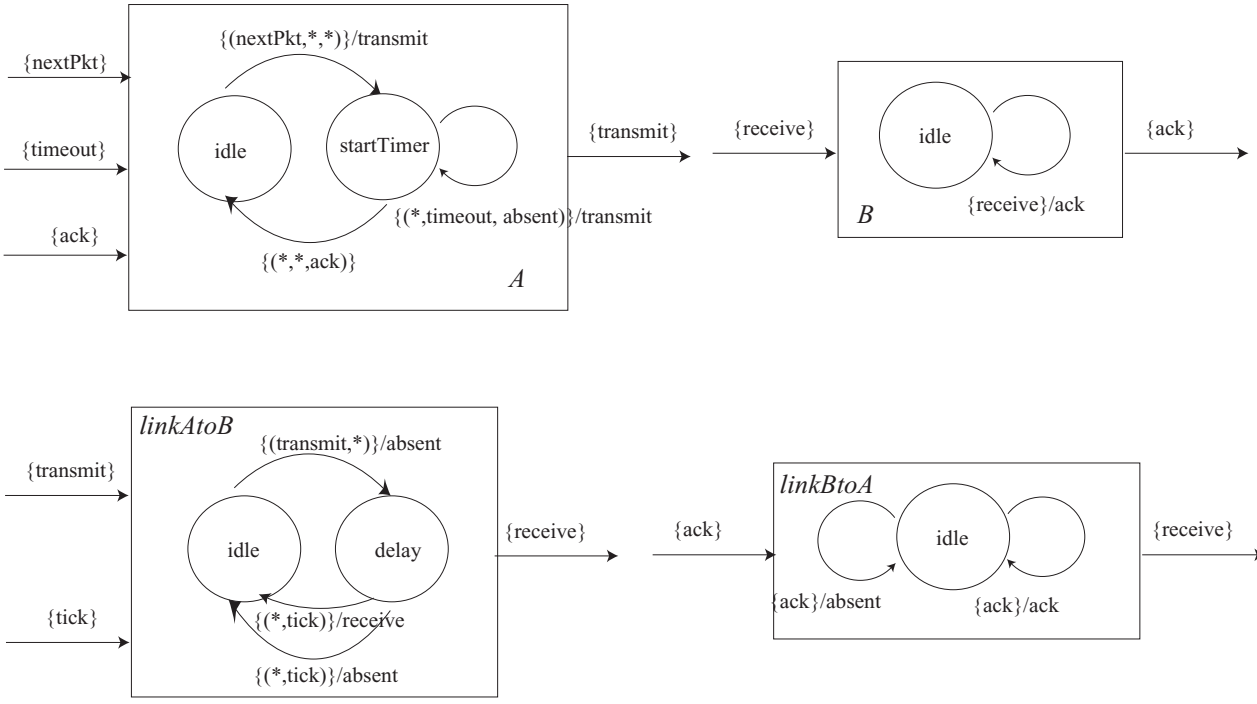
14. Only the *update* function needs to be modified. It becomes

$$\begin{aligned} \textit{possibleUpdates}((s_A, s_B), (x_A, x_B)) \quad = \quad & \{((s'_A, s'_B), (y_A, y_B)) \mid \\ & (s'_A, y_A) \in \textit{possibleUpdates}_A(s_A, x_A) \\ \wedge \quad & (s'_B, y_B) \in \textit{possibleUpdates}_B(s_B, x_B)\} \end{aligned}$$

15. Only the *update* function needs to be modified to

$$\begin{aligned} \textit{possibleUpdates}((s_A, s_B), x) \quad = \quad & \{((s'_A, s'_B), y_B) \mid \\ & \exists y'_A \text{ such that } (s'_A, y'_A) \in \textit{possibleUpdates}_A(s_A, x) \\ \wedge \quad & (s'_B, y'_B) \in \textit{possibleUpdates}_B(s_B, y_B)\} \end{aligned}$$

16. (a) The two machines A and B are shown on the top in this figure



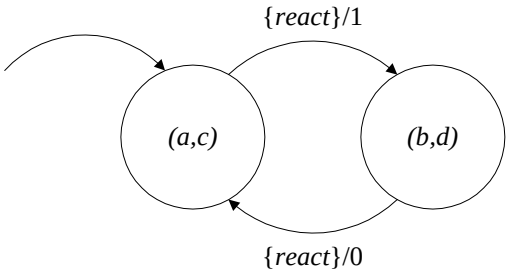
(b) The non-deterministic machines *linkAtoB* and *linkBtoA* model the forward and reverse links. *linkBtoA* has only one state, but *linkAtoB* has two states, one of which is a delay. The reason for adding the 'delay' state is to make the composition machine well-formed. (You can check for yourself that if *linkAtoB* was a one-state machine like *linkBtoA*, the composition would not be well-formed. *linkAtoB* is non-deterministic: in state *delay*, a *tick* may lead to output *receive*, which implies that *B* receives the packet or to output *absent*, which means that the packet is lost. Similarly, in *linkBtoA* an *ack* input may be forwarded to *A* or it may be lost.

(c) The composed state machine has $2 \times 2 \times 1 \times 1 = 4$ states. Each state is a four-tuple. However, since *B* and *linkBtoA* have only one state, we can suppress the corresponding coordinates as is done in the figure below. The input to the composite machine is in product form with three factors. The output is taken to be the output of *linkBtoA*. (We could have made a different choice.)

(d) If the machine is in state (*startTimer*,*delay*) and *timeout* occurs before *tick*, the machine will first make a self-loop transition and then transition to to (*startTimer*,*idle*).

17. (a) A and C are deterministic.

(b) The diagram is shown below.



(c)

$$\textit{Behaviors}_C = \{(a, b) \mid a = (\textit{react}, \textit{react}, \cdots) \ \wedge \ b = (1, 0, 1, 0, \cdots)\}$$

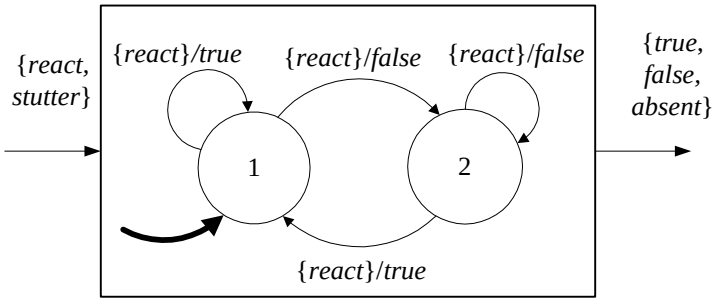
18. One solution is:

$$\begin{aligned} \textit{States} &= \{1, 2\} \\ \textit{Inputs} &= \{\textit{react}, \textit{stutter}\} \\ \textit{Outputs} &= \{\textit{true}, \textit{false}, \textit{absent}\} \\ \textit{initialState} &= 1 \end{aligned}$$

The *possibleUpdates* function is such that:

$$\begin{aligned} \textit{possibleUpdates}(1, \textit{react}) &= \{(1, \textit{true}), (2, \textit{false})\} \\ \textit{possibleUpdates}(2, \textit{react}) &= \{(1, \textit{true}), (2, \textit{false})\} \\ \textit{possibleUpdates}(1, \textit{stutter}) &= \{(1, \textit{absent})\} \\ \textit{possibleUpdates}(2, \textit{stutter}) &= \{(2, \textit{absent})\} \end{aligned}$$

A state transition diagram is shown below:



Note that this machine is bisimilar to one with only one state, and it produces an arbitrary sequence of *true* and *false*.

Chapter 5

Linear Systems

1. Given that

$$\boxed{\begin{array}{l} f \text{ is } \textbf{linear} \text{ if } \forall u, v \in \mathit{Reals}^N \text{ and } \forall a, b \in \mathit{Reals}, \\ f(au + bv) = af(u) + bf(v) \end{array}} \tag{5.1}$$

   then obviously the following is true for $n = 2$,

$$f(a_1u_1 + \cdots + a_nu_n) = a_1f(u_1) + \cdots + a_nf(u_n). \tag{5.2}$$

   To do induction, we assume it is true for some n and then show that it is true for $n + 1$. So we need to show that

$$f(a_1u_1 + \cdots + a_nu_n + a_{n+1}u_{n+1}) = a_1f(u_1) + \cdots + a_nf(u_n) + a_{n+1}f(u_{n+1}).$$

   We can use (5.1) to assert that

$$f((a_1u_1 + \cdots + a_nu_n) + a_{n+1}u_{n+1}) = f(a_1u_1 + \cdots + a_nu_n) + a_{n+1}f(u_{n+1}).$$

   The first term on the right can be expanded using (5.2), from which the result follows.

2. First, note that $A^0 = I$, by definition, which agrees with the hypothesized result. Now assume the hypothesized result is true for some $n >= 0$. We need to show that it is true for $n + 1$. To do this, note that

$$A^{n+1} = A^nA = \begin{bmatrix} a^n & na^{n-1} \\ 0 & a^n \end{bmatrix} \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix} = \begin{bmatrix} a^{n+1} & na^n + a^n \\ 0 & a^{n+1} \end{bmatrix},$$

   which agrees with the hypothesized result with n replaced by $n + 1$.

3. The composition $f \circ f$ is given by

$$\forall\, x \in \mathit{Reals}^N, \quad f \circ f(x) = A^2x.$$

   It is also linear.

4. In (5.27) we substitute $n = 32, a = 1.015b = 1, s(0) = -10000, s(32) = 0$ and $x(0) = \cdots = x(31) = w$ to get

$$0 = -1.015^{32} \times 10000 + \sum_{m=0}^{31} 1.015^{31-m} w,$$

from which

$$w = 0.015 \times \frac{10000 \times 1.015^{32}}{1.015^{32} - 1} = 395.7.$$

The total payment over 32 months is $395.7 \times 32 = 12665$.

5.

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad d = 0.$$

6. (a)

$$\begin{aligned} s(0) &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ s(1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned}$$

Since the state does not change, we can conclude it will never change while the input is zero, so for all $n \in$ *Integers*,

$$s(n) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

(b)

$$\begin{aligned} s(0) &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ s(1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ s(2) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\ s(3) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \end{aligned}$$

We conclude that for all $n \in$ *Integers*,

$$s(n) = \begin{bmatrix} n \\ 1 \end{bmatrix}.$$

(c) Following the same tactic as in (b), we conclude that for all $n \in$ *Integers*,

$$s(n) = \begin{bmatrix} n+1 \\ 1 \end{bmatrix}.$$

7. We can write

$$\begin{aligned} s(1) &= a + x(0) \\ s(2) &= a + x(0) + x(1) \\ s(3) &= a + x(0) + x(1) + x(2). \end{aligned}$$

Since $y(3) = s(3)$, to get $y(3) = 0$ we need

$$a + x(0) + x(1) + x(2) = 0.$$

With the three samples of x constrained to be equal to a constant b, then obviously $b = -a/3$.

8. Since $s(0) = [0, 0]^T$,

$$s(1) = As(0) + bx(0) = [0, x(0)]^T.$$

and

$$s(2) = As(1) + bx(1) = [x(0), x(1)]^T.$$

Thus, we need $x(0) = 1$ and $x(1) = 2$.

9. (a)

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 1/3 \\ 1/3 \end{bmatrix}, \quad d = 1/3.$$

(b)

$$A = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 1/3 \\ 0 \end{bmatrix}, \quad d = 1/3.$$

10. (a) If $\sigma = 0$, then the state will always be zero after the first reaction, so for all $n \in$ *Naturals*,

$$s(n) = [\, c\, 0\, 0 \,].$$

(b) The following Matlab script sets things up:

```
>> sigma = 0.9;
>> A = [cos(pi/6), sin(pi/6); -sin(pi/6), cos(pi/6)]

A =

    0.8660    0.5000
   -0.5000    0.8660

>> s(:,1) = [1;0]

s =

     1
     0
```

Notice the syntax of the last command. The colon says that all rows of the first column are being defined. This first column will represent the initial state $s(0)$. Unfortunately, Matlab arrays are indexed starting with 1, not 0, so we have to mentally do the translation. We can calculate the state $s(1)$ and store it in the second column as follows:

```
>> s(:,2) = A*s(:,1)

s =

    1.0000    0.8660
         0   -0.5000
```

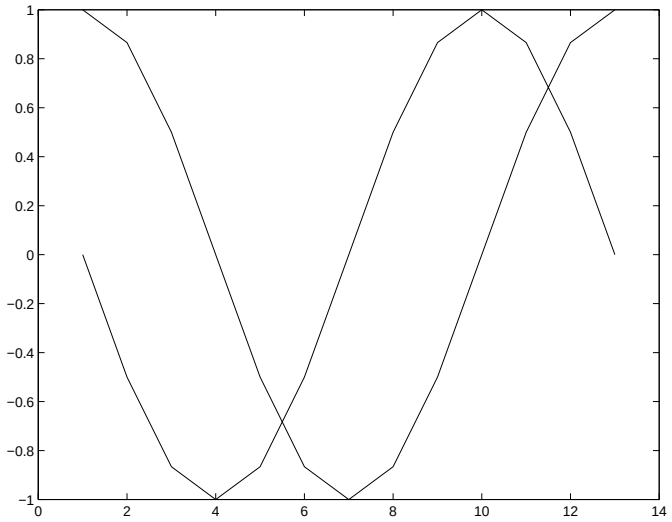
We can calculate the states $s(2), \cdots, s(12)$ as follows:

```
>> for i=3:13
      s(:,i) = A*s(:,i-1);
   end
>> s

s =

  Columns 1 through 7

1.0000   0.8660   0.5000   0.0000  -0.5000  -0.8660  -1.0000
     0  -0.5000  -0.8660  -1.0000  -0.8660  -0.5000  -0.0000

  Columns 8 through 13

-0.8660  -0.5000  -0.0000   0.5000   0.8660   1.0000
 0.5000   0.8660   1.0000   0.8660   0.5000   0.0000
```

We can plot the first and second elements of $s(n)$ versus n as follows:

```
>> plot(s')
```

getting

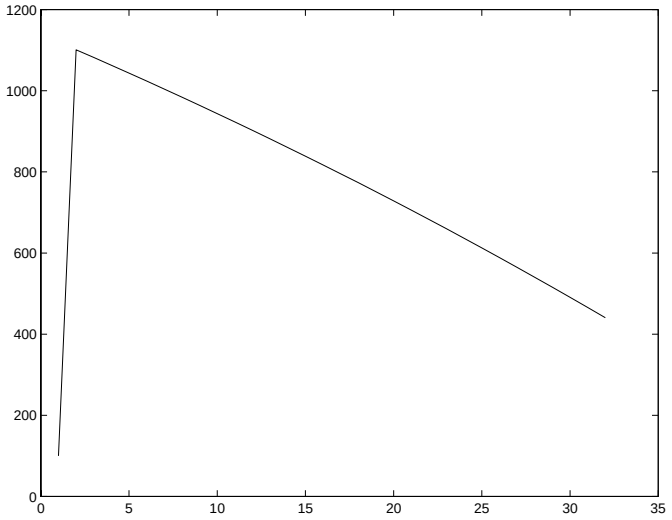


Note that the system behaves as an oscillator.

11. (a) Here we may use the computational steps given in the text. The only difference is that Matlab arrays are indexed starting with 1 rather than zero. Define $z(i) = s(i-1)$, and calculate as follows:

```
z(1) = 100;
z(2) = 1.01*z(1) + 1000;
for i=2:31
  z(i+1) = 1.01*z(i) - 30;
end
```

The plot is:



(b) From part (a) we find that the balance on day 31 is given by $s(32) = 440.4349$. We may

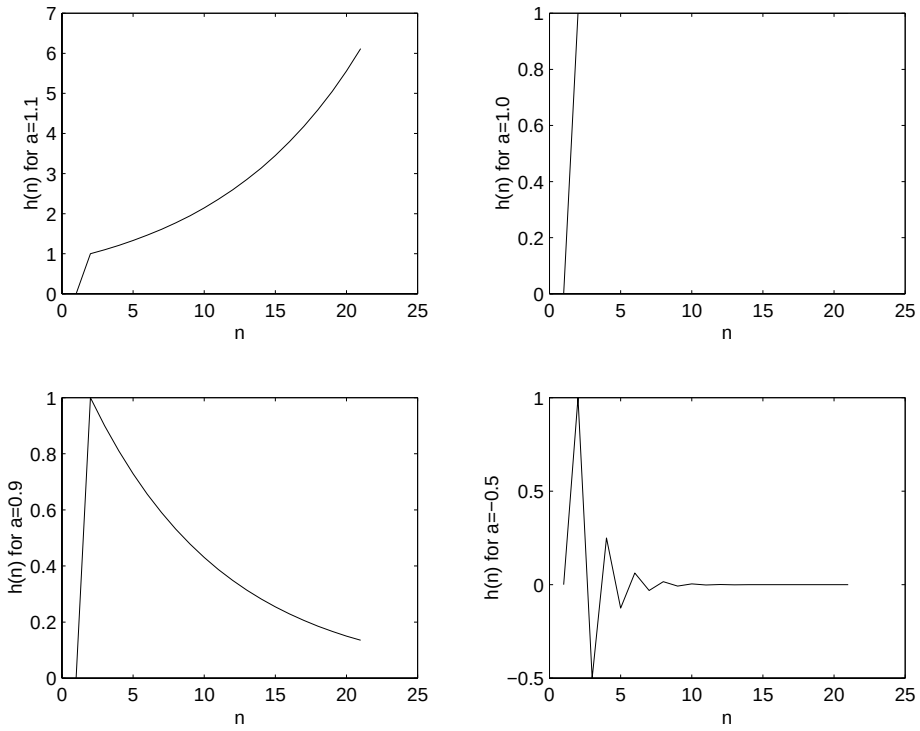
also find this result in closed form using the supplied identity:

$$\begin{aligned} s(31) &= (1.01)^{31}100 + \textstyle\sum_{m=0}^{30}(1.01)^{30-m}x(m) \\ &= (1.01)^{31}100 + (1.01)^{30}1000 - \textstyle\sum_{m=1}^{30}(1.01)^{30-m}30 \\ &= (1.01)^{31}100 + (1.01)^{30}1000 - 30(1.01)^{30}\textstyle\sum_{m=1}^{30}(1.01)^{-m} \\ &= (1.01)^{31}100 + (1.01)^{30}1000 - 30(1.01)^{30}\left(\frac{1-(1.01)^{31}}{1-1.01} - 1\right) \\ &= 440.4349 \end{aligned}$$

12. The impulse response is given by

$$h(n) = \begin{cases} d, & \text{if } n = 0 \\ ca^{n-1}b, & \text{if } n \geq 1 \end{cases} .$$

The impulse response for $a = 1.1$, $a = 1.0$, $a = 0.9$ and $a = -0.5$, where in all cases $b = c = 1$ and $d = 0$, is shown below:



13. For $n = 0$ (5.37) gives

$$s(0) = A^0 \textit{initialState} = \textit{initialState},$$

which is correct (note: $A^0 = I$, the identity matrix). Now suppose the right-hand side of (5.37) gives the correct value of $s(n)$ for some $n \geq 0$. From (5.33),

$$\begin{aligned} s(n+1) &= As(n) + bx(n) \\ &= A\{A^n \textit{initialState} + \sum_{m=0}^{n-1} A^{n-1-m}bx(m)\} + bx(n) \end{aligned}$$

$$= \quad A^{n+1}\textit{initialState} + \sum_{m=1}^{n} A^{n-m} b x(m),$$

which is the expression on the right-hand side of (5.37) for $n+1$. It follows by induction that (5.37) is the correct expression for the state response $s(n)$ for all $n \geq 0$. The fact that the output response is given by (5.37) now follows by substituting (5.37) in (5.34).

14. We want to show that for all $n = 0, 1, 2, \ldots$

$$A^n = \begin{bmatrix} \cos(n\omega) & -\sin(n\omega) \\ \sin(n\omega) & \cos(n\omega) \end{bmatrix}.$$

We may do so by induction. For $n = 0$:

$$A^0 = \begin{bmatrix} \cos 0 & -\sin 0 \\ \sin 0 & \cos 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

as required. Thus, the assertion is true for $n = 0$. Now, suppose the assertion is true for $n = k$. Then, for $k+1$:

$$\begin{aligned} A^{k+1} &= A^k A \\ &= \begin{bmatrix} \cos(k\omega)\cos(\omega) - \sin(k\omega)\sin(\omega) & -\cos(k\omega)\sin(\omega) - \sin(k\omega)\cos(\omega) \\ \sin(k\omega)\cos(\omega) + \cos(k\omega)\sin(\omega) & -\sin(k\omega)\sin(\omega) + \cos(k\omega)\cos(\omega) \end{bmatrix} \\ &= \begin{bmatrix} \cos((k+1)\omega) & -\sin((k+1)\omega) \\ \sin((k+1)\omega) & \cos((k+1)\omega) \end{bmatrix}. \end{aligned}$$

where the last equality follows from the given identities. This concludes the proof and the assertion is true for all $n = 0, 1, 2, \ldots$.

15. (a) The *zero-input state response* is given by:

$$\begin{aligned} s_{zero-input}(n) &= A^n \; initialState \\ &= \begin{bmatrix} \alpha^n \cos(n\omega) & -\alpha^n \sin(n\omega) \\ \alpha^n \sin(n\omega) & \alpha^n \cos(n\omega) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -\alpha^n \sin(n\omega) \\ \alpha^n \cos(n\omega) \end{bmatrix} \end{aligned}$$

and the *zero-input response* is given by:

$$\begin{aligned} y_{zero-input}(n) &= c^T A^n \; initialState \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} -\alpha^n \sin(n\omega) \\ \alpha^n \cos(n\omega) \end{bmatrix} \\ &= -\alpha^n \sin(n\omega). \end{aligned}$$

(b) The *zero-state impulse response* is given by:

$$h(n) = \begin{cases} d, & \text{if } n = 0 \\ c^T A^{n-1} b, & \text{if } n \geq 1 \end{cases}.$$

For the values of A, b, c in this problem we obtain:

$$h(n) = \begin{cases} 0, & \text{if } n = 0 \\ -\alpha^n \sin((n-1)\omega), & \text{if } n \geq 1 \end{cases}.$$

16. (a) A suitable model is:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 0 & 0 & \alpha & \beta \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \\ C &= \begin{bmatrix} 0 & 0 & 0 & \alpha & \beta \end{bmatrix}, \quad D = \begin{bmatrix} 1 \end{bmatrix}, \end{aligned}$$

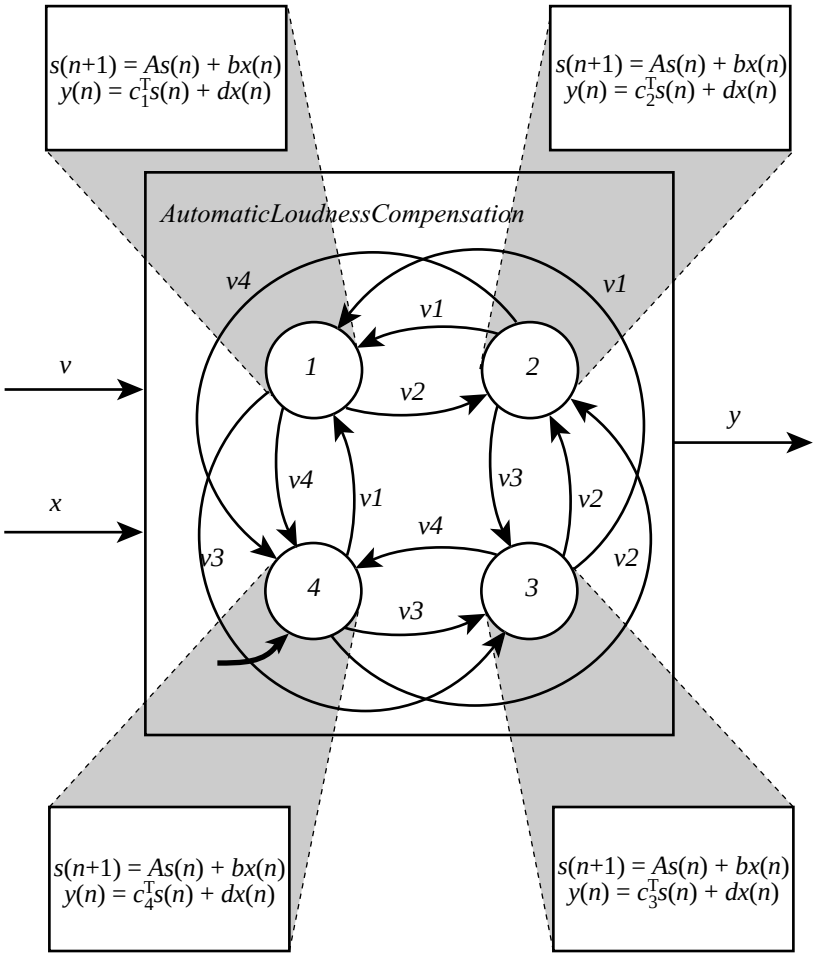
(b) The only items in the state-space representation that change are B and D, which become

$$B = \begin{bmatrix} 0.5 & 0.5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad D = [0.5 \;\; 0.5].$$

Chapter 6

Hybrid Systems – Solutions

1. The following hybrid system will do the job:



The inputs are two discrete-time signals, v representing the current volume setting, and x representing the audio signal. Each state i has a refinement that defines the state update and

output as

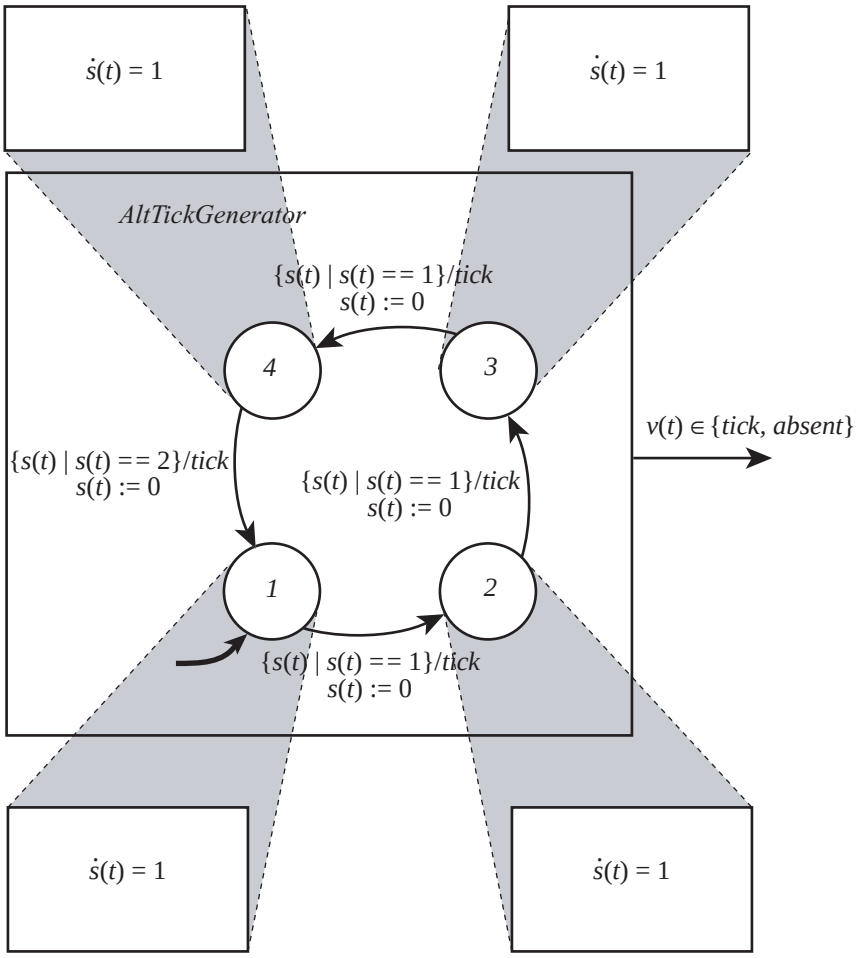
$$\begin{aligned} \forall\, n \in \textit{Integers}, \quad s(n+1) &= As(n) + bx(n) \\ y(n) &= c^T s(n) + dx(n). \end{aligned}$$

The guard on each transition into state i is a set $v_i \subset \textit{Inputs}$ given by

$$v_i = \{(v(n), x(n), s(n), y(n)) \mid T_{i-1} \leq v(n) < T_{i+1}\},$$

where $T_0, \cdots, T_5$ are thresholds governing the levels where the filtering switches. $T_0 = 0$ is the lowest level and $T_5 = \infty$ is the highest.

2. The following hybrid system will do the job:



Notice that all states have the same refinement, and that all transitions except the one into mode 1 have the same guard. The transition into mode 1 results in a delay of two units before the production of a *tick* event.

3. (a) The system generates an event sequence

$$(0, 1, 3, 4, 6, 7, 9, 10, \cdots)$$

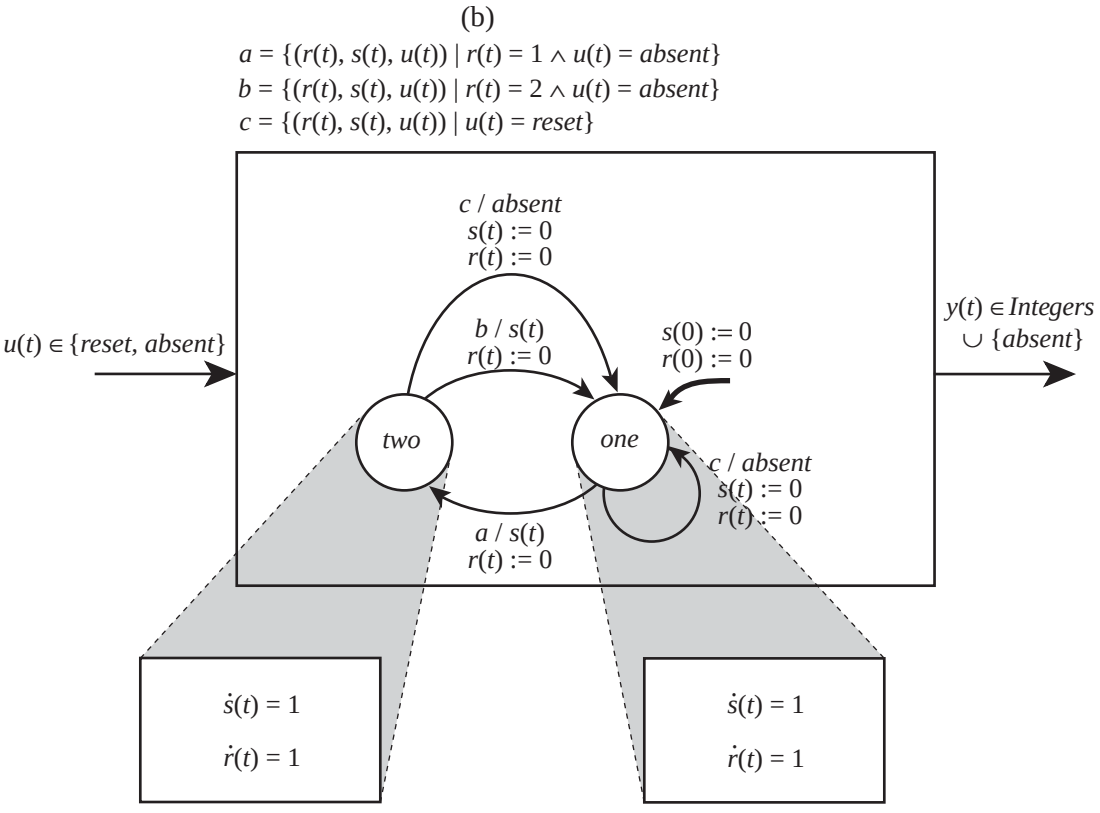
at times

$$0, 1, 3, 4, 6, 7, 9, 10, \cdots .$$

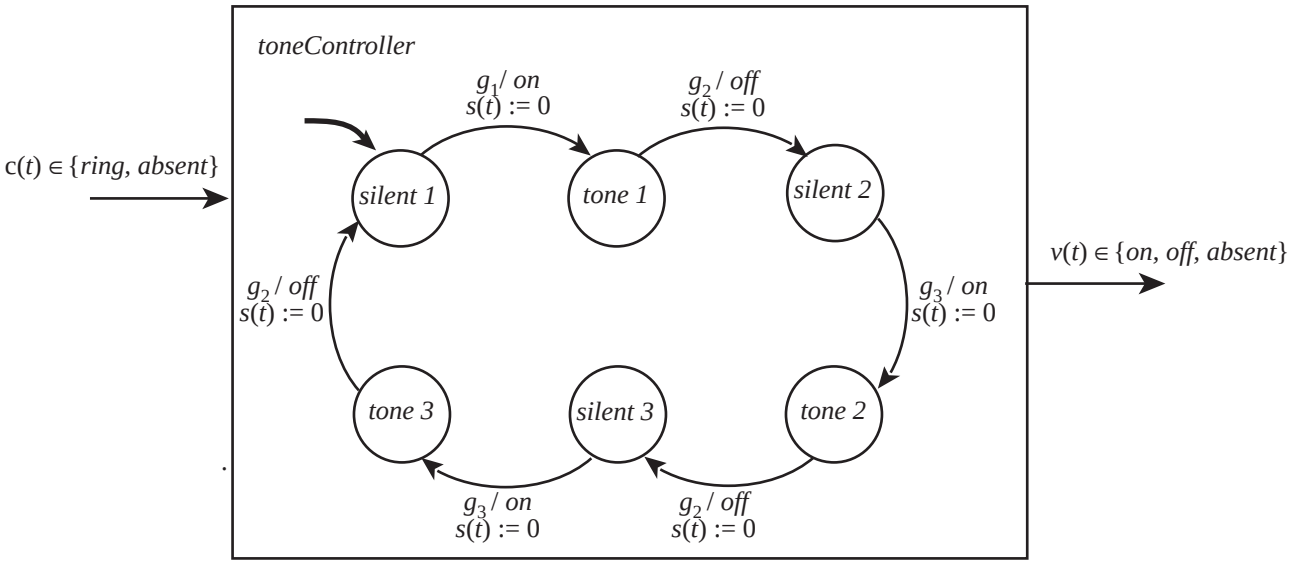
That is, the value of each output event is equal to the time at which it is produced, and the intervals between events alternate between one and two seconds. Precisely,

$$y(t) = \begin{cases} t & \text{if } t = 3k \text{ for some } k \in \mathit{Naturals}_0 \\ t & \text{if } t = 3k + 1 \text{ for some } k \in \mathit{Naturals}_0 \\ \mathit{absent} & \text{otherwise} \end{cases}$$

(b)



4. Assume the input alphabet is $\{\mathit{ring}, \mathit{absent}\}$ and the output alphabet is $\{\mathit{on}, \mathit{off}, \mathit{absent}\}$. Then the following timed automaton will control the source of the tone:



All states except *silent 1* have as a refinement the system given by

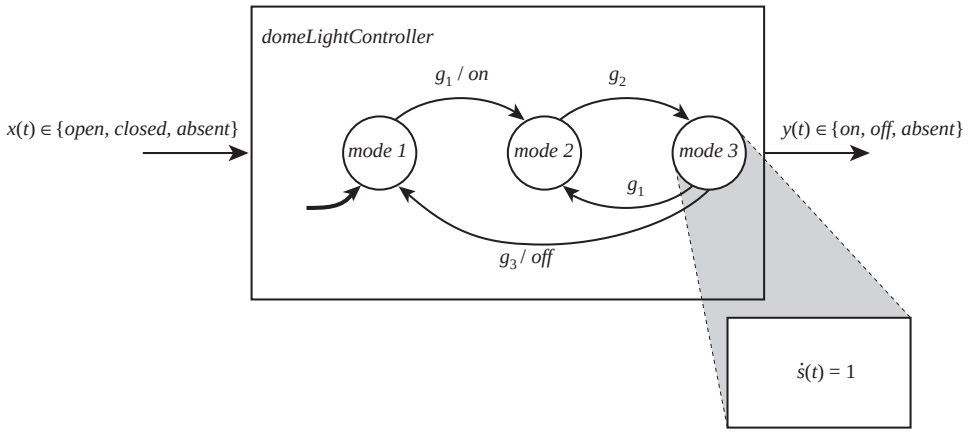
$$\dot{s}(t) = 1.$$

The guards are given by

$$\begin{aligned} g_1 &= \{(c(t), s(t)) \mid c(t) = \textit{ring}\} \\ g_2 &= \{(c(t), s(t)) \mid s(t) = 20\} \\ g_3 &= \{(c(t), s(t)) \mid s(t) = 10\}. \end{aligned}$$

This system ignores a *ring* input event that occurs less than 80 ms after the previous *ring* event.

5. (a) Assume that the automobile provides the input *open* when the first door is opened and *closed* when the last open door is closed. The following machine provides *on* to turn on the dome light and *off* to turn it off:



The guards are given by

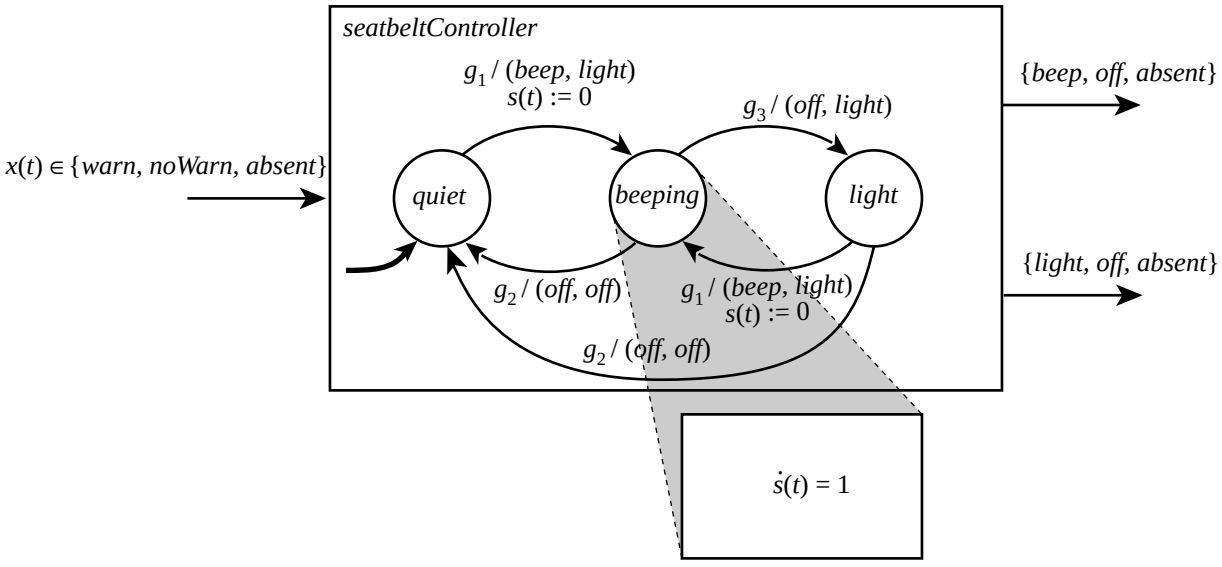
$$\begin{aligned} g_1 &= \{(x(t), s(t)) \mid x(t) = \textit{open}\} \\ g_2 &= \{(x(t), s(t)) \mid x(t) = \textit{closed}\} \\ g_3 &= \{(x(t), s(t)) \mid s(t) = 30\}. \end{aligned}$$

Such sensors can be easily implemented as a **daisy chain**, a series connection of switches so that if any door is open, the electrical circuit is open.

(b) Assume that the vehicle sensors provide the following input alphabet,

$$\textit{Inputs} = \{\textit{warn}, \textit{noWarn}, \textit{absent}\},$$

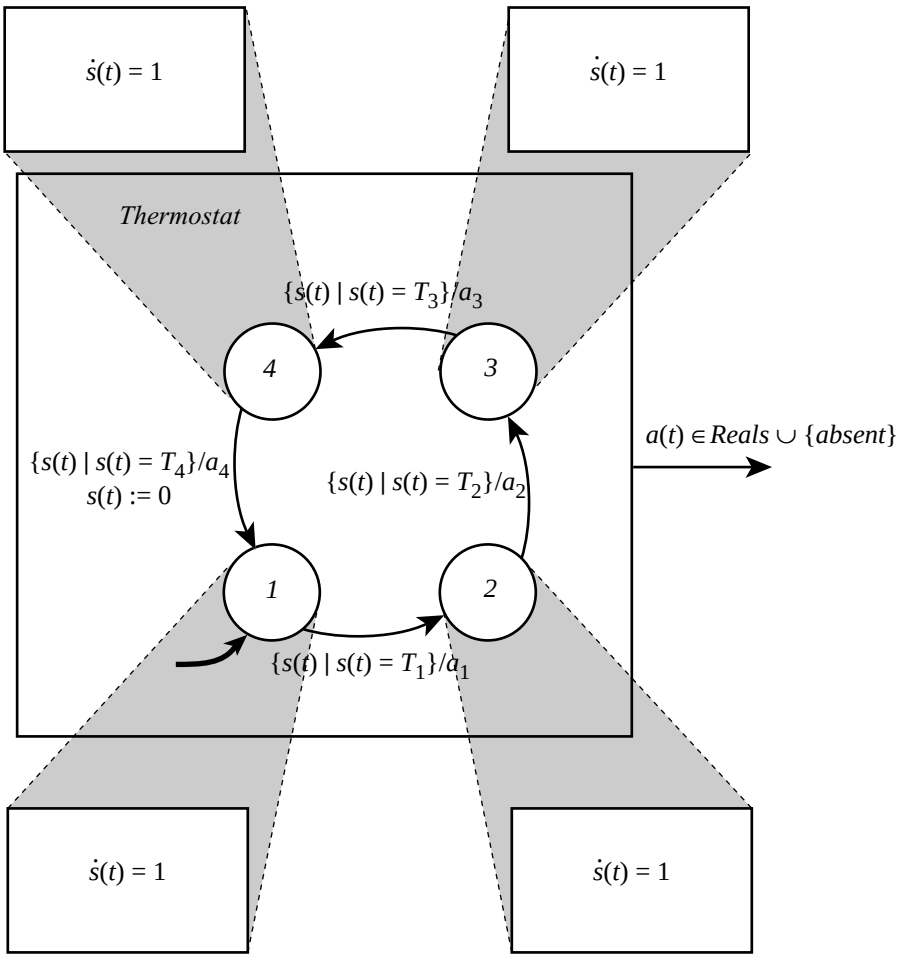
as suggested in the hint. The following model provides the requisite control:



The guards are given by

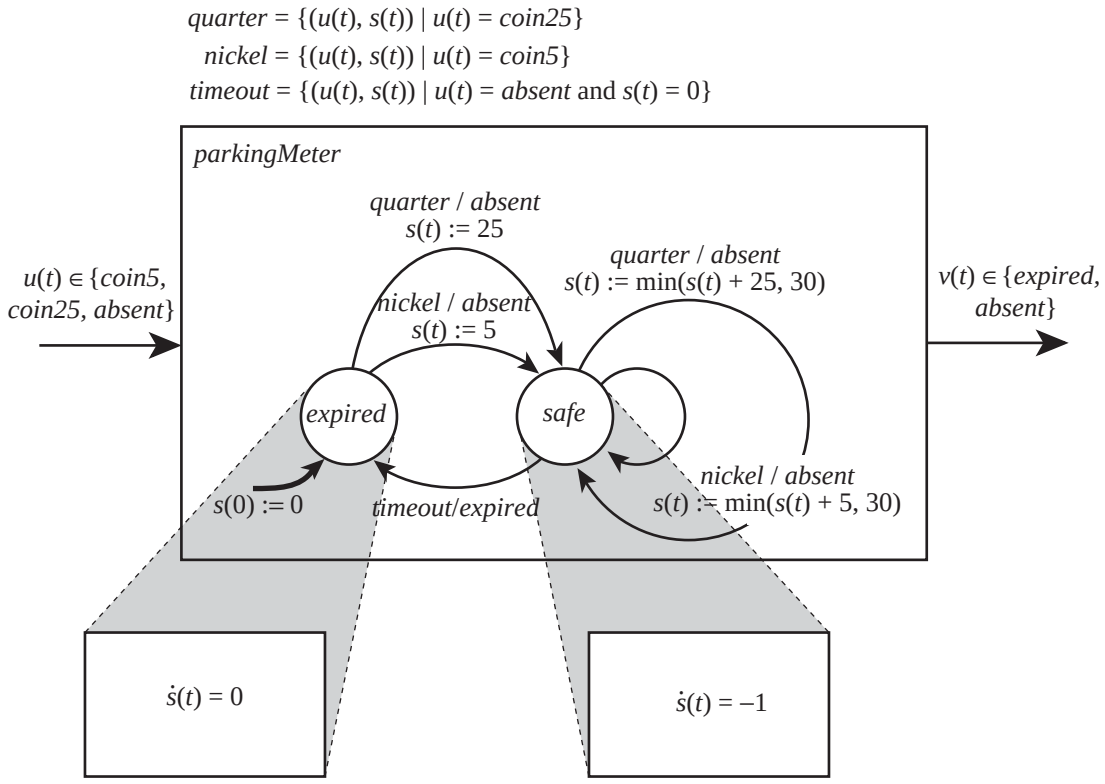
$$\begin{aligned} g_1 &= \{(x(t), s(t)) \mid x(t) = \textit{warn}\} \\ g_2 &= \{(x(t), s(t)) \mid x(t) = \textit{noWarn}\} \\ g_3 &= \{(x(t), s(t)) \mid s(t) = 30 \ \wedge \ x(t) = \textit{absent}\}. \end{aligned}$$

6. The following hybrid system will do the job:

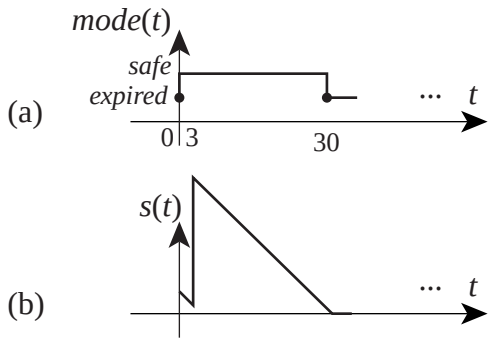


You need four modes and one timer.

7. The hybrid system is almost identical, with just 60 replaced with 30:

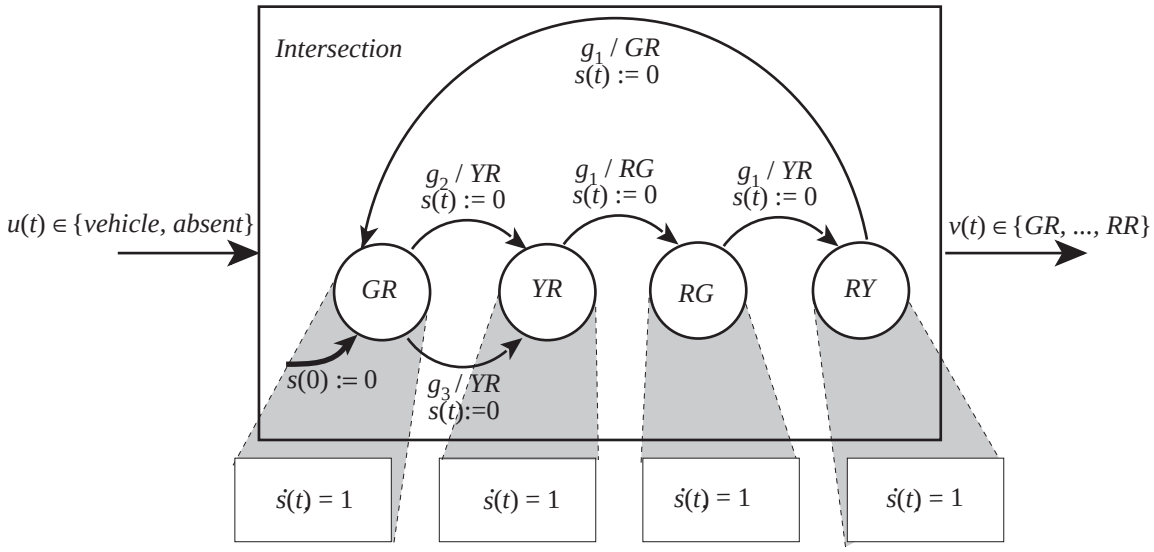


The required state trajectory is shown below:



8. The answer is no. The behavior cannot be reproduced by a finite state machine. The input symbols to the machine are in the alphabet $\{coin5, coin25, absent\}$. At each reaction, one of these inputs must be presented. If the stuttering input, *absent*, is presented, then the machine must stutter, and the output must be *absent*. But then the only other two possible inputs are *coin5* and *coin25*. Neither of these will result in output *expired* in the parking meter, so a state machine with these inputs and outputs would never produce the output symbol *expired*.

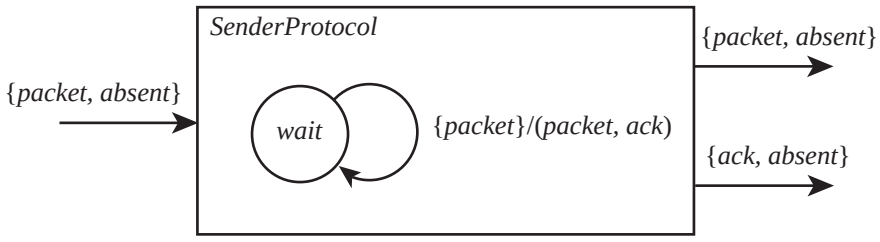
9. The hybrid system is described below. Time t is in seconds. Four modes are needed corresponding to the four possible light configurations.



The guards are given by

$$g_1 = \{(u(t), s(t)) \mid s(t) = 20\}$$
$$g_2 = \{(u(t), s(t)) \mid s(t) = 240 \ \wedge \ u(t) \neq \textit{vehicle}\}$$
$$g_3 = \{(u(t), s(t)) \mid u(t) = \textit{vehicle}\}.$$

10. The receiver protocol is a one-state finite state machine with one input from its NIC and two outputs: if a packet is received it is forwarded to the receiver application program, and at the same time an *ack* packet is sent to the sender. The machine is shown below.



There is a 'bug' in this protocol. Suppose the sender sends a packet which is correctly received. The *ReceiverProtocol* forwards the packet to the receiver application and sends an *ack*. Suppose the *ack* arrives *after* the sender's timer expires, in which case it will retransmit the same packet, and suppose this duplicate packet is also received. In that case the duplicate will be forwarded to the receiver, which would be wrong. To prevent forwarding duplicates, in internet protocols, the sender also attaches a unique sequence number to each packet. The *ReceiverProtocol* can now keep track of this sequence number and discard duplicate packets.

11. (a) Suppose $v(n)$ is the velocity just before the nth bounce. Then $v(n+1) = av(n)$ and so $v(n) = a^{n-1}v(1)$.

    (b) The time $t_n - t_{n-1}$ is the time taken by the ball to return to the ground, starting at velocity $v(n)$. So $t_n - t_{n-1} = 2v(n)/g$.

    (c) The maximum height reached after the $(n-1)$st bounce is $\frac{1}{2}\frac{(v(n))^2}{g}$.

12. (a) We need to write

$$\ddot{y_1}(t) = k_1(p_1 - y_1(t))/m_1 \tag{6.1}$$
$$\ddot{y_2}(t) = k_2(p_2 - y_2(t))/m_2. \tag{6.2}$$

    as

$$\dot{s}(t) = g(s(t)) \tag{6.3}$$

    where

$$s(t) = \begin{bmatrix} y_1(t) \\ \dot{y_1}(t) \\ y_2(t) \\ \dot{y_2}(t) \end{bmatrix}.$$

    Here is one way to write this:

$$\dot{s}(t) = \begin{bmatrix} \dot{y_1}(t) \\ \ddot{y_1}(t) \\ \dot{y_2}(t) \\ \ddot{y_2}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -k_1/m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_2/m_2 & 0 \end{bmatrix} \begin{bmatrix} y_1(t) \\ \dot{y_1}(t) \\ y_2(t) \\ \dot{y_2}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ k_1p_1/m_1 \\ 0 \\ k_2p_2/m_2 \end{bmatrix}.$$

    Because of the non-zero added term on the end, the function g is not linear.

    (b) We need to write

$$\ddot{y}(t) = -g$$

    as

$$\dot{s}(t) = f(s(t))$$

    where

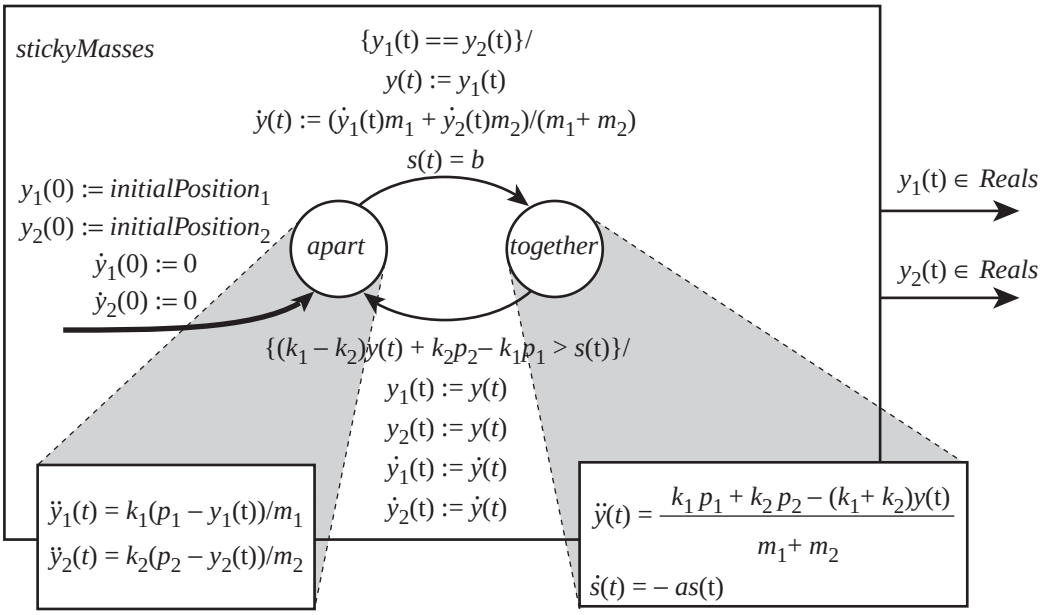
$$s(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}.$$

    Here is one way to write this:

$$\dot{s}(t) = \begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -g \end{bmatrix}.$$

    Because of the non-zero added term on the end, the function f is not linear.

13. The model is shown below



14. In *left* mode, the refinement state evolves according to

$$\begin{aligned} \dot{x}(t) &= 10\cos\phi(t) \\ \dot{y}(t) &= 10\sin\phi(t) \\ \dot{\phi}(t) &= \pi \end{aligned}$$

Intuitively this suggests that the vehicle moves in a circle with speed 10 mph. By integrating these differential equations we get

$$\begin{aligned} x(t) - x(0) &= \int_0^t \cos(\phi(0) + \pi t)dt = \frac{10}{\pi}\sin(\phi(0) + \pi t) \\ y(t) - y(0) &= \int_0^t \sin(\phi(0) + \pi t)dt = -\frac{10}{\pi}\cos(\phi(0) + \pi t) \\ \phi(t) - \phi(0) &= \pi t \end{aligned}$$

So $(x(t) - x(0))^2 + (y(t) - y(0))^2 = \frac{10^2}{\pi^2}$, which means the vehicle is moving in a circle of radius $10/\pi$, and it takes 2 seconds to complete a circle.

15. We call the hybrid system *AGV*,

$$AGV = (States, Inputs, Outputs, TransitionStructure, initialState),$$

where

$$\begin{aligned} States &= Modes \times RefinementStates \\ Modes &= \{stop, right, straight, left\} \\ RefinementStates &= Reals^3 = \{s = (x, y, \phi) \mid x, y, \phi \in Reals\} \\ Inputs &= InputEvents = \{stop, right, straight, left, absent\}, \\ Outputs &= OutputEvents = \{absent\} \\ initialState &= (stop, x_0, y_0, \phi_0) \end{aligned}$$

AGV has no time-based inputs or outputs. We now describe the *TransitionStructure* for two modes, *stop* and *right*, the others are similar. Call $u(t)$ the input and $e(t)$ the displacement at time t. ($e(t)$ is a function of the refinement state $(x(t), y(t), \phi(t))$.)

If state at t is $(\textit{stop}, s(t))$ and

$$\begin{array}{rl} \text{if guard } \{u(t) = \textit{straight}\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{straight}, s(t)) \\ \text{if guard } \{u(t) = \textit{right}\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{right}, s(t)) \\ \text{if guard } \{u(t) = \textit{left}\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{left}, s(t)) \end{array}$$

else if $T_{\textit{stop}}$ is the set of times t when the mode is *stop*, then $\forall t \in T_{\textit{stop}}$,

$$\begin{aligned} \dot{x}(t) &= 0 \\ \dot{y}(t) &= 0 \\ \dot{\phi}(t) &= 0 \end{aligned}$$

If state at t is $(\textit{right}, s(t))$ and

$$\begin{array}{rl} \text{if guard } \{|e(t)| < \epsilon_1\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{straight}, s(t)) \\ \text{if guard } \{0 > -\epsilon_2 > e(t)\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{left}, s(t)) \\ \text{if guard } \{u(t) = \textit{stop}\} \text{ is true,} & \text{then state at } t+ \text{ is } (\textit{stop}, s(t)) \end{array}$$

else if $T_{\textit{right}}$ is the set of times t when the mode is *right*, then $\forall t \in T_{\textit{right}}$,

$$\begin{aligned} \dot{x}(t) &= 10 \cos \phi(t) \\ \dot{y}(t) &= 10 \sin \phi(t) \\ \dot{\phi}(t) &= -\pi \end{aligned}$$

Chapter 7

Frequency Domain Solutions

1. (a) A finite discrete-time signal is a signal whose domain is a some finite subset

$$[a, b] \subset \textit{Integers},$$

where $a, b \in$ *Integers* and $a < b$. A discrete-time signal y: *Integers* $\rightarrow$ *Reals* is periodic with period $p \in$ *Integers* if

$$\forall\, n \in \textit{Integers}, \quad y(n) = y(n + p).$$

As a consequence, $y(n) = y(n + kp)$, for all integers k.

(b) A finite image is a signal whose domain is a subset

$$[a, b] \times [c, d] \subset \textit{Reals} \times \textit{Reals}$$

for some finite a, b, c, d where $a < b$ and $c < d$. A periodic image will have to have an infinite domain. The signal

$$\textit{Image}\colon \textit{Reals} \times \textit{Reals} \rightarrow \textit{Intensity}$$

is periodic with horizontal period $p_h \in$ *Reals* and vertical period $p_v \in$ *Reals* if

$$\forall\, x, y \in \textit{Reals}, \quad \textit{Image}(x, y) = \textit{Image}(x + p_h, y) = \textit{Image}(x, y + p_v).$$

As a consequence,

$$\textit{Image}(x, y) = \textit{Image}(x + kp_h, y + mp_v)$$

for all integers k, m.

2. (a)

$$\forall\, t \in \textit{Reals}, \quad x(t) = 10 \sin(2\pi t) + (10 + 2i) \cos(2\pi t)$$

This function is periodic with a period of 1, since both components are periodic with that period.

(b) The signal x is the sum of two periodic signals, say x_1 with period p_1 (fundamental frequency $f_1 = 1/p_1$), and x_2 with period p_2 (fundamental frequency $f_2 = 1/p_2$). So x is periodic with period p if and only if there are positive integers k_1 and k_2 such that

$$p = k_1 p_1 = k_2 p_2.$$

The period of x is therefore the smallest p for which this relation holds. Note that if the relation holds, then

$$p_1/p_2 = f_2/f_1 = k_2/k_1,$$

which must be rational since k_1 and k_2 are both integers. Thus x given by

$$\forall\, t \in \textit{Reals}, \quad x(t) = \sin(2\pi t) + \sin(\sqrt{2}\pi t)$$

is not periodic since $f_1 = 1$, and $f_2 = \sqrt{2}/2$ are not rationally related.

(c) Using the same technique as in (b), the function x given by

$$\forall\, t \in \textit{Reals}, \quad x(t) = \sin(2\sqrt{2}\pi t) + \sin(\sqrt{2}\pi t)$$

is periodic. Here $f_1 = \sqrt{2}$ and $f_2 = \sqrt{2}/2$. Let $k_1 = 2$, $k_2 = 1$, and $f = \sqrt{2}/2 = 1/\sqrt{2}$. Therefore the signal is periodic with a period of $1/f = 1.414$.

3. (a) The period is $p = 9$, or any integer multiple of 9.

(b) With $p = 9$, the fundamental frequency is $f_0 = 1/9$ cycles/sample, or $\omega_0 = 2\pi/9$ radians/sample.

(c) With $p = 9$ we get $A_0 = A_2 = 1$, while all other A_i are 0. $\phi_2 = 0$, and all other ϕ_i are arbitrary.

4. (a) $p = 2$.

(b) $\omega_0 = \pi$ radians/second.

(c) $A_2 = A_3 = 1$, $A_k = 0$, $\forall\, k \notin \{2, 3\}$, and $\phi_k = 0$, $\forall\, k \in \textit{Naturals}$.

5. The fundamental frequency is the largest f_0 such that for each i, $f_i = k_i f_0$ for some integer k_i , where f_i is the frequency of the i-th component.

In this case, $f_1 = 440$ Hz, $f_2 = 554$ Hz, and $f_3 = 659$ Hz. Notice that f_3 is a prime number and that f_1 and f_2 are integers. Therefore, the largest f_0 that divides f_1, f_2, and f_3 into integers is 1 Hz. That is, the fundamental frequency is 1 Hz.

Since $\sin(x) = \cos(x - \pi/2)$, by inspection, the Fourier series coefficients are

$$\begin{aligned} A_{440} &= 1, \phi_{440} = -\pi/2 \\ A_{554} &= 1, \phi_{554} = -\pi/2 \\ A_{659} &= 1, \phi_{659} = -\pi/2 \end{aligned}$$

All other coefficients A_k, ϕ_k are 0.

6. Observe that

$$\cos(\omega_0 t + \phi) = Re\{e^{j\omega_0 t + \phi}\} = Re\{e^{j\omega_0 t} e^{j\phi}\}.$$

Writing each term on the left and right side of the equation in this form and factoring we determine that

$$ARe\{e^{j\omega_0 t} e^{j\phi}\} = 5Re\{e^{j\omega_0 t}(e^{j\pi/2} + e^{-j\pi/6} + e^{-j2\pi/3})\}.$$

We can calculate the sum of the three exponentials using Matlab:

```
>> s = exp(j*pi/2) + exp(-j*pi/6) + exp(-j*2*pi/3)

s =

   0.3660 - 0.3660i

>> angle(s)

ans =

   -0.7854

>> abs(s)*5

ans =

    2.5882
```

Hence, $A = 2.5882$ and $\phi = -0.7854 = -\pi/4$.

7. (a) A radio wave travels at the speed of light, c m/s. Furthermore, as it travels, the wave gets attenuated (because its power spreads in three dimensional space). So if x is the transmitted radio wave, the radio wave received at a distance l meters away at time t will be $y(t) = ax(t - l/c)$, where a is the attenuation $(a < 1)$.

   (b) If there were 10 reflected paths (one direct, 9 reflected), the received signal would be $\forall\, t \in$ *Reals*,

$$y(t) = A \sum_{k=0}^{9} \alpha_k \cos(2\pi f(t - l_k/c))$$

   where l_k is the length of the k-th path and is the attenuation along that path.

   (c) The phase shift along the k-th path is

$$\phi_k = 2\pi f l_k / c,$$

   which has units of radians. Notice that the wavelength is $\lambda = c/f$, so

$$\phi_k = 2\pi l_k / \lambda,$$

(d) Since $\phi_k - \phi_0 = 2\pi f(l_k - l_0)$,

$$\Phi = 2\pi L f/c.$$

(e) In this case,

$$y(t) = A\alpha_0 \cos(2\pi f t - \phi_0) + A\alpha_1 \cos(2\pi f t - \phi_0 - \pi).$$

Note that in general $\cos(\theta - \pi) = -\cos(\theta)$, so

$$y(t) = A\cos(2\pi f t - \phi_0)(\alpha_0 - \alpha_1).$$

So the amplitude of the direct signal is reduced by the full amount of the amplitude of the reflected signal. The reflected signal is said to be 180 degrees out of phase.

(f) Using the result of (d), to get $\Phi \leq \pi/10$ we need

$$2\pi L f/c \leq \pi/10.$$

Since $L \leq 500$, we need to constrain f so that

$$2 \times 500 f/c \leq 1/10.$$

Using $c = 3 \times 10^8$, we need

$$f \leq 3 \times 10^4.$$

So if we can use a radio frequency below 30 kHz, then we are assured of not having much destructive interference. Unfortunately, there is not much bandwidth below 30 kHz, and it is impractical to build mobile radios at these frequencies anyway because of the huge antennas that are required. Thus, mobile radio systems have to contend with destructive multipath interference.

(g) A frequency interferes destructively if

$$\phi_1 - \phi_0 = 2\pi f L/c = (2n+1)\pi.$$

for some integer n. I.e.,

$$L = (2n+1)c/2f = (2n+1)\lambda/2.$$

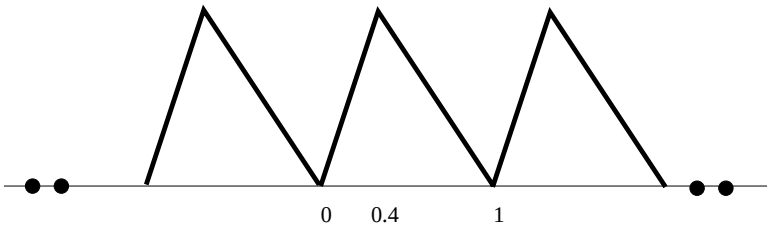
Thus, we get destructive interference if L equals an odd multiple of half the wavelength.

8. (a) For any $t \in$ *Reals*,

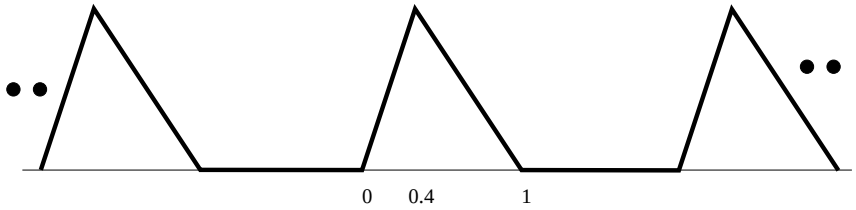
$$y(t+p) = \sum_{k=-\infty}^{\infty} x(t+p-kp) = \sum_{m=-\infty}^{\infty} x(t-mp) = y(t),$$

by change of variables, $m = k - 1$. Hence, y is periodic with period p.

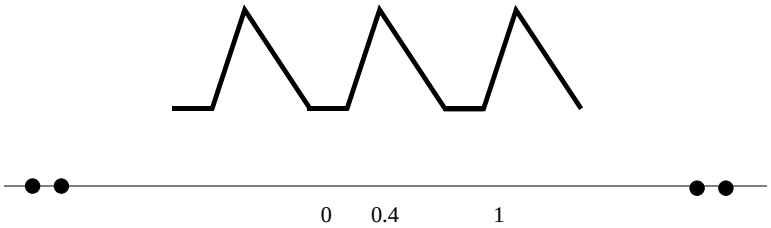
(b) With $p = 1$, the graph of y looks like this:



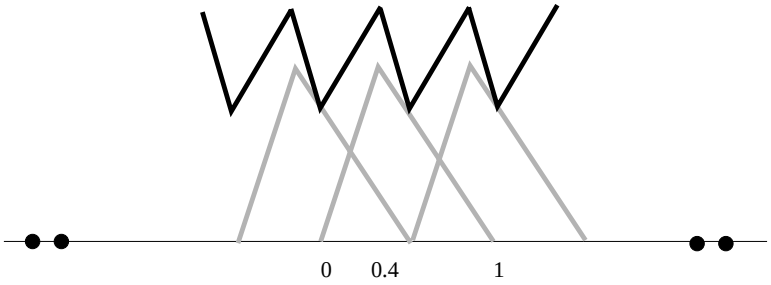
(c) With $p = 2$, the graph of y looks like this:



(d) With $p = 0.8$, the graph of y looks like this:



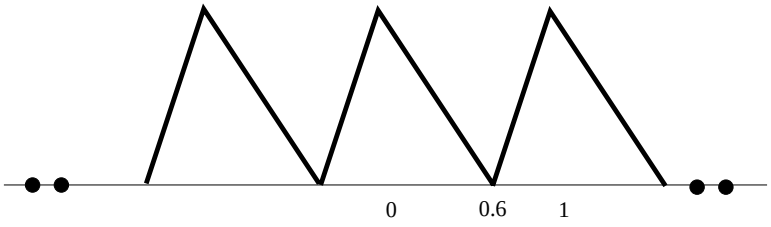
(e) With $p = 0.5$, the graph of y looks like this:



(f) In this case,

$$w(t) = y(t + 4).$$

With $p = 1$, the graph of w looks like this:



9. (a) We need to show that for all $t \in$ *Reals*, $y(t) = y(t + p)$. But this is true, since

$$y(t + p) = f(x(t + p)) = f(x(t)) = y(t)$$

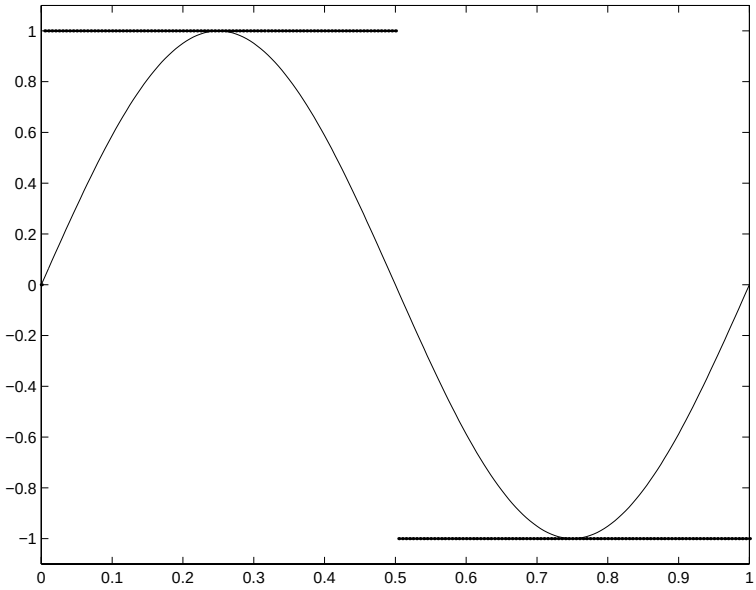


Figure 7.1: Solution to (b) of problem 9.

since x is periodic with period p.

(b) The plot over one period is shown in figure 7.1. It was generated using the following Matlab code:

```
t = [0:1/200:1];
x = sin(2*pi*t);
y = sign(x);
plot(t,x,'-',t,y,'.');
```

(c) The plot over one period of x is shown in figure 7.2. It was generated using the following Matlab code:

```
t = [0:1/200:1];
x = sin(2*pi*t);
y = x.*x;
plot(t,x,'-',t,y,'.');
```

10. Denote the square wave by s_p: *Reals* $\rightarrow$ *Reals*. Assume the periodic image is a function

$$\textit{Image}: \textit{Reals} \times \textit{Reals} \rightarrow \textit{Reals}$$

given by $\forall\ (x, y) \in$ *Reals* $\times$ *Reals*,

$$\textit{Image}(x, y) = s_h(x)s_v(y).$$

Thus, given that

$$s_p(t) = A_0 + \sum_{k=1}^{\infty} A_k \cos(2\pi kt/p + \phi_k)$$

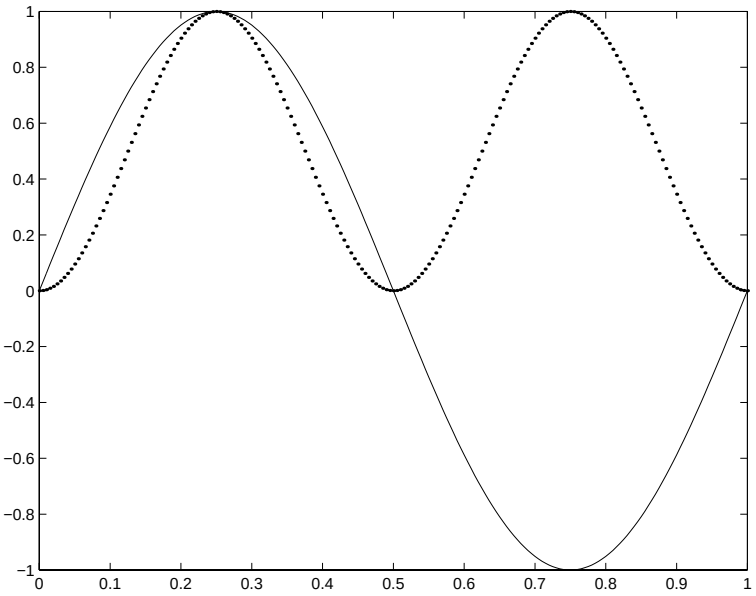


Figure 7.2: Solution to (c) of problem 9.

we have

$$\textit{Image}(x,y) = [A_0 + \sum_{k=1}^{\infty} A_k \cos(2\pi kx/p + \phi_k)][A_0 + \sum_{k=1}^{\infty} A_k \cos(2\pi ky/p + \phi_k)].$$

If we let $\phi_0 = 0$ and change variables on the second summation, this can be written

$$\textit{Image}(x,y) = [\sum_{k=0}^{\infty} A_k \cos(2\pi kx/p + \phi_k)][\sum_{m=0}^{\infty} A_m \cos(2\pi my/p + \phi_m)].$$

Multiplying out the summations yields

$$\textit{Image}(x,y) = \sum_{k=0}^{\infty}\sum_{m=0}^{\infty} A_k A_m \cos(2\pi kx/p + \phi_k)\cos(2\pi my/p + \phi_m),$$

from which we recognize that the Fourier series coefficients for the image are

$$A_{k,m} = A_k A_m,$$

with phases ϕ_k and $\varphi_m = \phi_m$.

11. Evaluating both sides at $t = 0$, it follows immediately that

$$A_1 = A_2.$$

Taking derivatives on both sides, we get that

$$\forall\, t \in \textit{Reals} \quad i\omega_1 A_1 e^{i\omega_1 t} = i\omega_2 A_2 e^{i\omega_2 t}.$$

Divide both sides by i and evaluate at $t = 0$ to get

$$\omega_1 A_1 = \omega_2 A_2.$$

Since $A_1 = A_2$, it must also be true that

$$\omega_1 = \omega_2.$$

Chapter 8

Frequency Response Solutions

1. Let $A = \alpha e^{i\theta}$ (in polar coordinates), so

$$\begin{aligned} Ae^{i\omega t} + A^* e^{-i\omega t} &= 2Re\{\alpha e^{i\omega t + \theta}\} \\ &= 2\alpha\cos(\omega t + \theta). \end{aligned}$$

   From this, it is evident that $\alpha = 1/2$ and $\theta = \pi/4$. Hence,

$$A = 0.5e^{i\pi/4}.$$

2. Note that

$$s(x) = Im\{e^{(-x+i2\pi x)}\} = e^{-x}\sin(2\pi x)$$

   which is plotted in figure 8.1. This plot was generated by the following Matlab code:

```
x = [-1:0.01:1];
s = exp(-x).*sin(2*pi*x);
plot(x,s)
```

3. ϕ in terms of τ, ω is $\phi = \tau\omega$. τ has units of seconds. ω has units of radians per second. ϕ has units of radians.

4. By definition, x is periodic with period p if for all $t \in$ *Reals*,

$$x(t + p) = x(t).$$

   The left side here is simply $(D_p(x))(t)$. Therefore, an equivalent definition of periodic with period p would be that

$$D_p(x) = x.$$

   Since S is time invariant,

$$S(D_p(x)) = D_p(S(x)).$$

   The left side is $S(x)$ because $D_p(x) = x$. Hence

$$D_p(S(x)) = S(x),$$

   so $S(x)$ is periodic with period p.

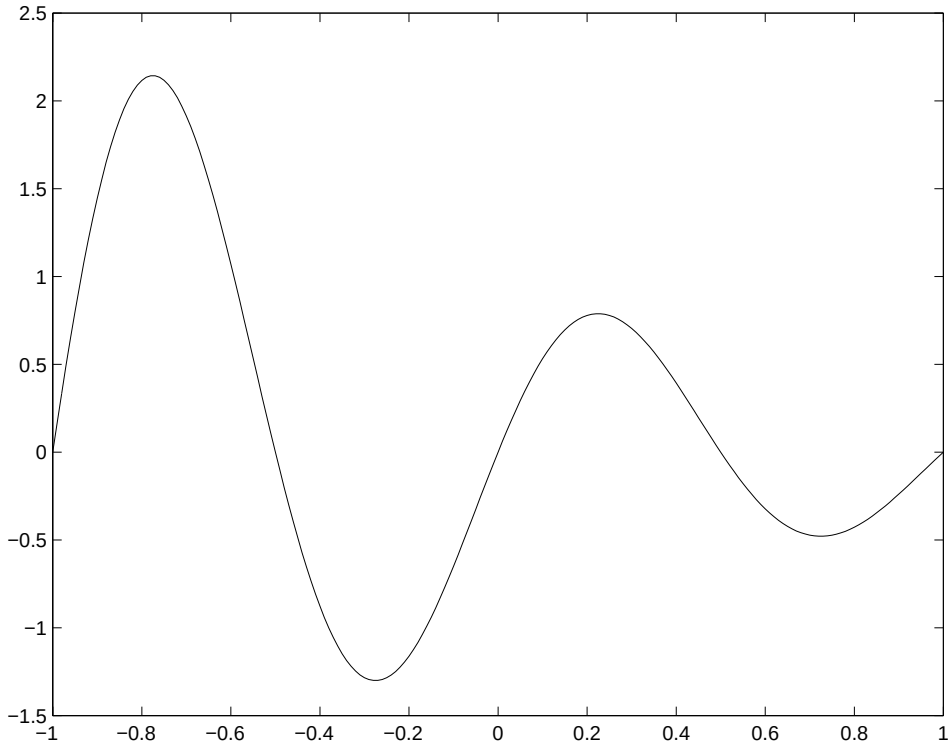


Figure 8.1: Solution to problem 2.

5. (a) Assume *Images* $= [$*Reals* $\times$ *Reals* $\rightarrow Y]$, where Y is intensities, colormap indexes, or colors. Then $S_{v,h}$: *Images* $\rightarrow$ *Images* such that for all $I \in$ *Images* and $\alpha, \beta \in$ *Reals*,

$$(D_{v,h}(I))(\alpha, \beta) = I(\alpha - v, \beta - h).$$

(b) Assume *Images* $= [$*Integers* $\times$ *Integers* $\rightarrow Y]$, where Y is intensities, colormap indexes, or colors. Then $S_{m,n}$: *Images* $\rightarrow$ *Images* such that for all $I \in$ *Images* and $\alpha, \beta \in$ *Integers*,

$$(D_{m,n}(I))(\alpha, \beta) = I(\alpha - m, \beta - n).$$

6. (a) Yes, it is linear. Check superposition. If $y_1 = D(x_1)$ and $y_2 = D(x_2)$, then

$$\begin{aligned}(D(ax_1 + bx_2))(n) &= ax_1(n-1) + bx_2(n-1) \\ &= a(D(x_1))(n) + b(D(x_2))(n),\end{aligned}$$

so superposition applies.

(b) Yes, it is time invariant. Note simply that

$$D_m \circ D = D \circ D_m = D_{m+1}.$$

7. (a) Yes, it is linear. Let $y_1 =$ *TimeScale*(x_1), and $y_2 =$ *TimeScale*(x_2). I.e.,

$$y_1(t) = x_1(2t) \quad \text{and} \quad y_2(t) = x_2(2t).$$

Let $y =$ *TimeScale*$(ax_1 + bx_2)$. I.e.,

$$y(t) = ax_1(2t) + bx_2(2t).$$

Observe from this latter equation that

$$y = a \cdot \textit{TimeScale}(x_1) + b \cdot \textit{TimeScale}(x_2),$$

so superposition applies.

(b) No, it is not time invariant. Let

$$y_1 = (D_\tau \circ \textit{TimeScale})(x)$$

and

$$y_2 = (\textit{TimeScale} \circ D_\tau)(x).$$

I.e.,

$$y_1(t) = x(2t - \tau)$$

and

$$y_2(t) = x(2t - 2\tau).$$

These are not equal for all x, so the system is not time invariant.

8. Write x in terms of complex exponentials:

$$x(t) = e^{i0t} + 0.5e^{i\pi t} + 0.5e^{-i\pi t} + 0.5e^{i2\pi t} + 0.5e^{-i2\pi t},$$

and note that each term gets scaled by the frequency response,

$$y(t) = H(0)e^{i0t} + 0.5H(\pi)e^{i\pi t} + 0.5H(-\pi)e^{-i\pi t} + 0.5H(2\pi)e^{i2\pi t} + 0.5H(-2\pi)e^{-i2\pi t}.$$

Then note that $H(0) = 1$, $H(\pi) = -1$, $H(-\pi) = -1$, $H(2\pi) = 0$, and $H(-2\pi) = 0$, so

$$y(t) = 1 - \cos(\pi t).$$

9. (a) $A'_k = aA_k,\ \forall\ k \in \textit{Naturals}_0$ and $\phi'_k = \phi_k,\ \forall\ k \in \textit{Naturals}$.
   (b) $A'_k = A_k,\ \forall\ k \in \textit{Naturals}_0$ and $\phi'_k = \phi_k - k\omega_0\tau,\ \forall\ k \in \textit{Naturals}$.
   (c) $A'_0 = A_0$, $A'_k = 0,\ \forall\ k \in \textit{Naturals}$ and $\phi'_k = 0,\ \forall\ k \in \textit{Naturals}$. (Any other value for ϕ'_k is acceptable.
   (d)

$$\forall\ k \in \textit{Naturals}_0, \quad A'_k = |A_k e^{i\phi_k} + A''_k e^{i\phi''_k}|,$$

   and

$$\forall\ k \in \textit{Naturals}, \quad \phi'_k = \angle(A_k e^{i\phi_k} + A''_k e^{i\phi''_k}).$$

10. (a) LTI
    (b) LTI
    (c) L
    (d) L
    (e) TI
    (f) TI

11. Note that

$$x(n) = e^{i\omega n}$$

where $\omega = 0$. Therefore, $y = S(x)$ is given by

$$y(n) = H(\omega)e^{i\omega n} = H(0) \cdot 1 = |\sin(0)| = 0.$$

Hence,

$$y(n) = 0$$

for all $n \in \textit{Integers}$.

12. Note that the term being summed is a periodic function of k. Also note that it is a complex exponential. Summing a complex exponential over one period yields zero. The period is the smallest integer p such that

$$5(k+p)\pi/6 = 5k\pi/6 + M2\pi$$

for some integer M. Solving for p, we find

$$p = 12M/5.$$

Since both M and p must be integers, we find that the period is $p = 12$. Thus, $n = 11$ will result in summing over one period, which will result in zero.

We can verify this more directly. Let $\rho = e^{i5\pi/6}$. Then

$$\sum_{k=0}^{n} e^{i5k\pi/6} = \sum_{k=0}^{n} \rho^k = \frac{1-\rho^{n+1}}{1-\rho}.$$

This is zero if

$$1 - \rho^{n+1} = 0$$

or

$$\rho^{n+1} = 1.$$

So we need to find the smallest $n \in$ *Naturals* such that

$$\rho^{n+1} = e^{i5\pi(n+1)/6} = 1.$$

So we need to find the smallest $n \in$ *Naturals* such that

$$5\pi(n+1)/6$$

is a multiple of 2π. I.e., we need to find the smallest $n \in$ *Naturals* such that

$$5\pi(n+1)/6 = M2\pi$$

for some integer M, or

$$n + 1 = 12M/5$$

which means

$$n = 11.$$

13. (a) Using the relations

$$\begin{aligned} X_0 &= A_0, \\ X_k &= 0.5A_k e^{i\phi_k}, \quad k = 1, 2, \cdots \\ X_{-k} &= {X_k}^* = 0.5A_k e^{-i\phi_k}, \quad k = 1, 2, \cdots \end{aligned}$$

we get

$$X_k = \begin{cases} 1, & k = 0 \\ 0.5, & k \in \{-2, -1, 1, 2\} \\ 0, & \text{otherwise} \end{cases}$$

(b) Using the results of part (a), we can write

$$x(t) = e^{i0} + 0.5(e^{-i2t} + e^{-it} + e^{it} + e^{i2t}).$$

Because the system is LTI, the output is

$$y(t) = H(0)e^{i0} + 0.5(H(-2)e^{-i2t} + H(-1)e^{-it} + H(1)e^{it} + H(2)e^{i2t}).$$

Since $H(\omega) = \cos(\pi\omega/2)$, we have $H(0) = 1$, $H(1) = H(-1) = 0$, and $H(2) = H(-2) = -1$. Thus,

$$\begin{aligned} y(t) &= 0.5(-e^{-i2t} - e^{i2t}) + 1 \\ &= -cos(2t) + 1. \end{aligned}$$

(c) The fundamental frequency of the output is $\omega_0' = 2$ radians/second.

14. Note that

$$x'(t) = x(t) - x(t-1).$$

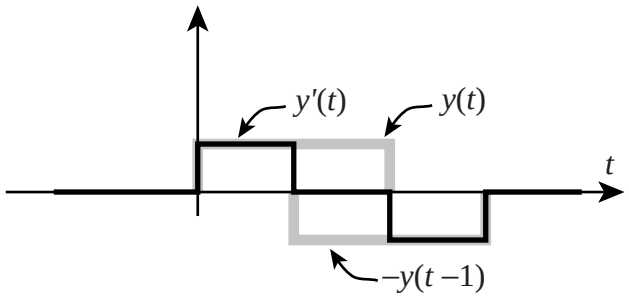
Since the system is LTI,

$$y'(t) = y(t) - y(t-1).$$

That is,

$$y(t) = \begin{cases} 1, & \text{if } 0 \le t < 1 \\ -1, & \text{if } 2 \le t < 3 \\ 0, & \text{otherwise} \end{cases}$$

A sketch is given below:



15. (a) From Euler's relation,

$$cos(\pi n/2) = 0.5(e^{i\pi n/2} + e^{-i\pi n/2}).$$

Since the system is LTI, the output is given by

$$\begin{aligned} y(n) &= 0.5(H(\pi/2)e^{i\pi n/2} + H(-\pi/2)e^{-i\pi n/2}) \\ &= 0.5((\pi/2)e^{i\pi n/2} + (\pi/2)e^{-i\pi n/2}) \\ &= 0.5\pi \cos(\pi n/2). \end{aligned}$$

(b) Write the input as

$$x(n) = 5e^{i0n}$$

so the output is recognized as

$$y(n) = 5H(0)e^{i0n} = 0.$$

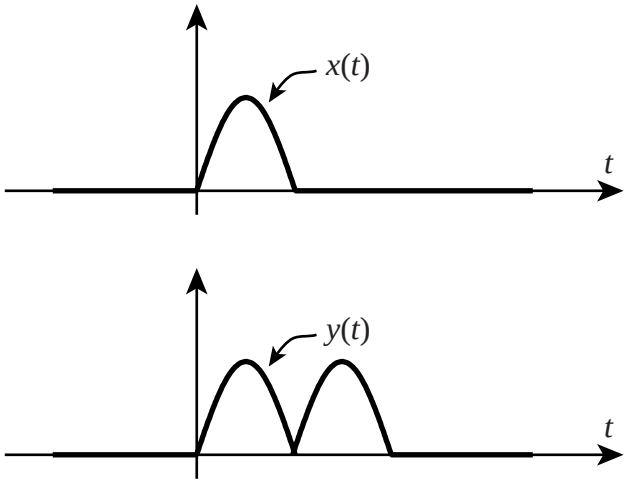
(c) Write the input as a complex exponential function,

$$x(n) = e^{i\pi n}$$

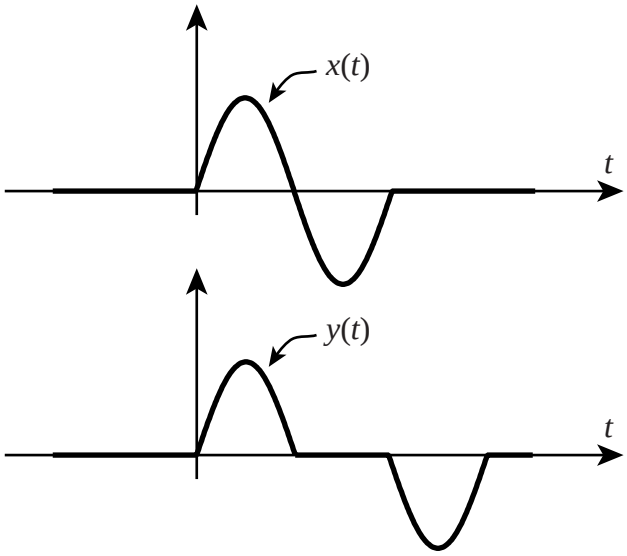
so the output is given by

$$\begin{aligned} y(n) &= H(\pi)e^{i\pi n} \\ &= \pi e^{i\pi n} \\ &= \begin{cases} \pi, & n \text{ even} \\ -\pi, & n \text{ odd} \end{cases} \end{aligned}$$

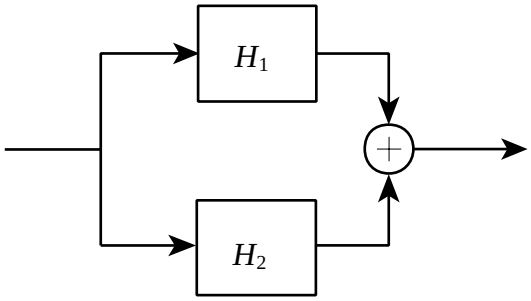
16. (a) The sketches are shown below:



(b) The sketches are shown below:



17. The system is shown below:



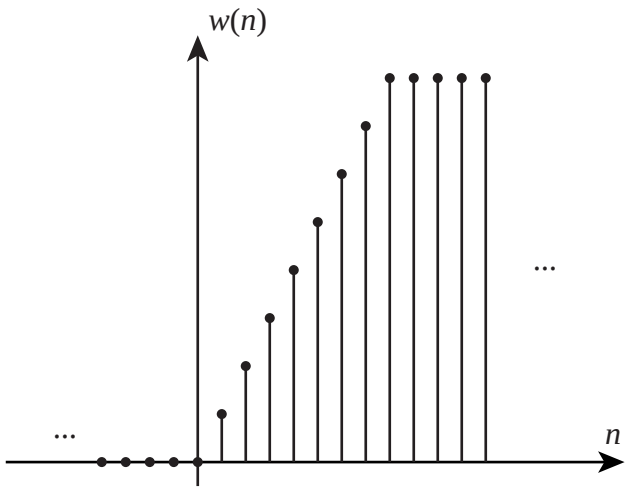
18. Note that

$$p(n) = 2(u(n) - u(n-8)).$$

Since the system is LTI,

$$\begin{aligned} w(n) &= 2(y(n) - y(n-8)) \\ &= 2(nu(n) - (n-8)u(n-8)) \\ &= \begin{cases} 2n & 0 \le n < 8 \\ 16 & n \ge 8 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A sketch is shown below



19. (a) Given

$$y(t) = x(at)$$

and

$$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{ik\omega_0 t}$$

it follows that

$$y(t) = \sum_{k=-\infty}^{\infty} X_k e^{ik\omega_0 at}.$$

Thus, the Fourier series coefficients for y are the same as those for x, but the fundamental frequency is $\omega_0 a$, where $\omega_0 = 2\pi/p$ is the fundamental frequency for x.

(b) Since

$$w(t) = x(t)e^{i\omega_0 t}$$

it follows that

$$\begin{aligned} w(t) &= \sum_{k=-\infty}^{\infty} X_k e^{i\omega_0 t} e^{ik\omega_0 t} \\ &= \sum_{k=-\infty}^{\infty} X_k e^{i(k+1)\omega_0 t}. \end{aligned}$$

Change variables, letting $m = k + 1$, to get

$$w(t) = \sum_{m=-\infty}^{\infty} X_{m-1} e^{im\omega_0 t}.$$

Therefore, the fundamental frequency for w is the same as for x, and its Fourier series coefficients are:

$$W_k = X_{k-1}$$

for all integers k.

(c) Write

$$\begin{aligned} z(t) &= x(t)\cos(\omega_0 t) \\ &= 0.5x(t)(e^{i\omega_0 t} + e^{-i\omega_0 t}) \\ &= 0.5\sum_{k=-\infty}^{\infty} X_k(e^{i\omega_0 t} + e^{-i\omega_0 t})e^{ik\omega_0 t} \\ &= 0.5\sum_{m=-\infty}^{\infty} X_{m-1}e^{im\omega_0 t} + 0.5\sum_{m=-\infty}^{\infty} X_{m+1}e^{im\omega_0 t}. \end{aligned}$$

Thus, once again the fundamental frequency is the same, and the Fourier series coefficients are

$$Z_k = 0.5(X_{k-1} + X_{k+1})$$

for all integers k.

20. To see the validity of (8.33),

$$X_k = \frac{1}{p}\sum_{m=0}^{p-1} x(m)e^{-imk\omega_0},$$

we substitute for $x(m)$ from (8.30) in the right-hand side to get

$$\frac{1}{p}\sum_{m=0}^{p-1}\{\sum_{n=0}^{p-1} X_n e^{inm\omega_0}\}e^{-imk\omega_0} = \frac{1}{p}\sum_{n=0}^{p-1} X_n \sum_{m=0}^{p-1} e^{im(n-k)\omega_0} = X_k,$$

because if $n \neq k$,

$$\sum_{m=0}^{p-1} e^{im(n-k)\omega_0} = \frac{1 - e^{i(n-k)p\omega_0}}{1 - e^{-i(n-k)\omega_0}} = 0,$$

and if $n = k$,

$$\sum_{m=0}^{p-1} e^{im(n-k)\omega_0} = p.$$

21. (a) We need to show that for all $x \in [\textit{Reals} \to \textit{Reals}]$ and $t \in \textit{Reals}$,

$$\textit{Squarer}(x))(t) = f(x(t))$$

for some function $f\colon \textit{Reals} \to \textit{Reals}$. Let f be such that for all $u \in \textit{Reals}$, $f(u) = u^2$. Then

$$f(x(t)) = x^2(t) = \textit{Squarer}(x))(t).$$

Hence the function is memoryless.

(b) To show that the system is not linear, it is sufficient to show that for some $a \in \textit{Reals}$ and $x \in [\textit{Reals} \to \textit{Reals}]$,

$$\textit{Squarer}(ax) \neq a\textit{Squarer}(x).$$

To show this, we can find any $t \in \textit{Reals}$ such that

$$(\textit{Squarer}(ax))(t) \neq a(\textit{Squarer}(x)(t)).$$

The left side equals $a^2x^2(t)$, while the right side is $ax^2(t)$. These are not equal if, for example, $a = 2$ and $x(t) \neq 0$.

(c) We need to show that

$$\textit{Squarer} \circ D_\tau = D_\tau \circ \textit{Squarer}.$$

Both sides define the same function f given by

$$\forall\, x \in [\textit{Reals} \to \textit{Reals}], t \in \textit{Reals}, \quad (f(x))(t) = x^2(t-\tau).$$

(d) Observe that

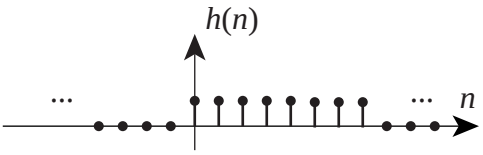
$$\begin{aligned} y(t) &= \cos^2(\omega t) \\ &= (0.5(e^{i\omega t} + e^{-i\omega t}))^2 \\ &= 0.25(e^{i2\omega t} + 1 + 1 + e^{-ie\omega t})^2 \\ &= 0.5 + 0.5\cos(2\omega t), \end{aligned}$$

from which we see that the output contains a constant term and a sinusoidal term at frequency 2ω.

Chapter 9

Filtering Solutions

1. (a) The impulse response is shown below:



   (b) When the input is given by $x(n) = \cos(\omega n)$, where $\omega = \pi/4$, then the output is $y(n) = 0$. From the figure showing the magnitude frequency response of the 8-point moving average, we see that $H(\pi/4) = 0$. Since the impulse response is real, $H(-\omega) = H^*(\omega)$, so $H(-\pi/4) = 0$ as well.

   (c) $H(\omega) = 0$.

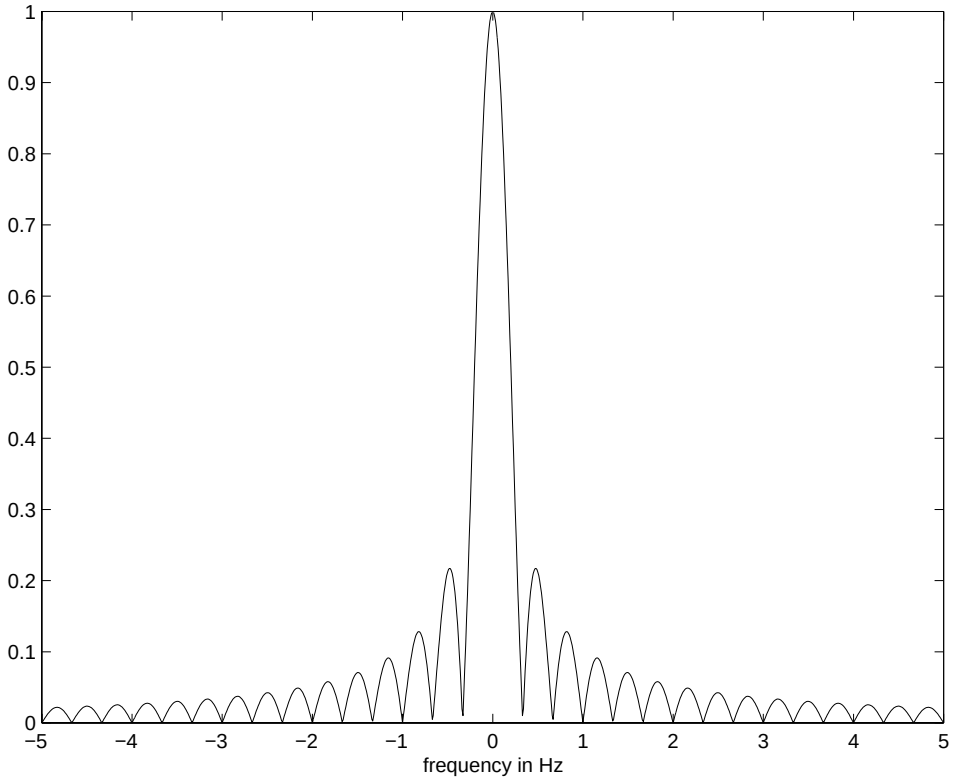
2. We can calculate the CTFT of the impulse response,

$$\begin{aligned} H(\omega) &= \int_{-\infty}^{\infty} h(t)e^{-i\omega t}dt \\ &= \int_{0}^{3} (1/3)e^{-i\omega t}dt \\ &= (1 - e^{-i3\omega})/(3i\omega). \end{aligned}$$

   The following Matlab code plots the magnitude response:

```
f = [-5:1/100:5];
H = (1-exp(-i*3*2*pi*f))./(3*i*2*pi*f);
plot(f,abs(H));
```

   Note that this gives a "Warning: Divide by zero" at frequency 0, but generates a correct plot anyway. You can use L'Hopital's rule to find that the value at frequency zero is 1. The plot is shown below:



3. (a) Using convolution,

$$\begin{aligned} y(t) &= \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \\ &= \int_{-\infty}^{\infty} (\delta(\tau-1)+\delta(\tau-2))x(t-\tau)d\tau \\ &= \int_{-\infty}^{\infty} \delta(\tau-1)x(t-\tau)d\tau + \int_{-\infty}^{\infty} \delta(\tau-2)x(t-\tau)d\tau \\ &= = x(t-1)+x(t-2), \end{aligned}$$

using the sifting rule.

(b) The frequency response is the CTFT of the impulse response,

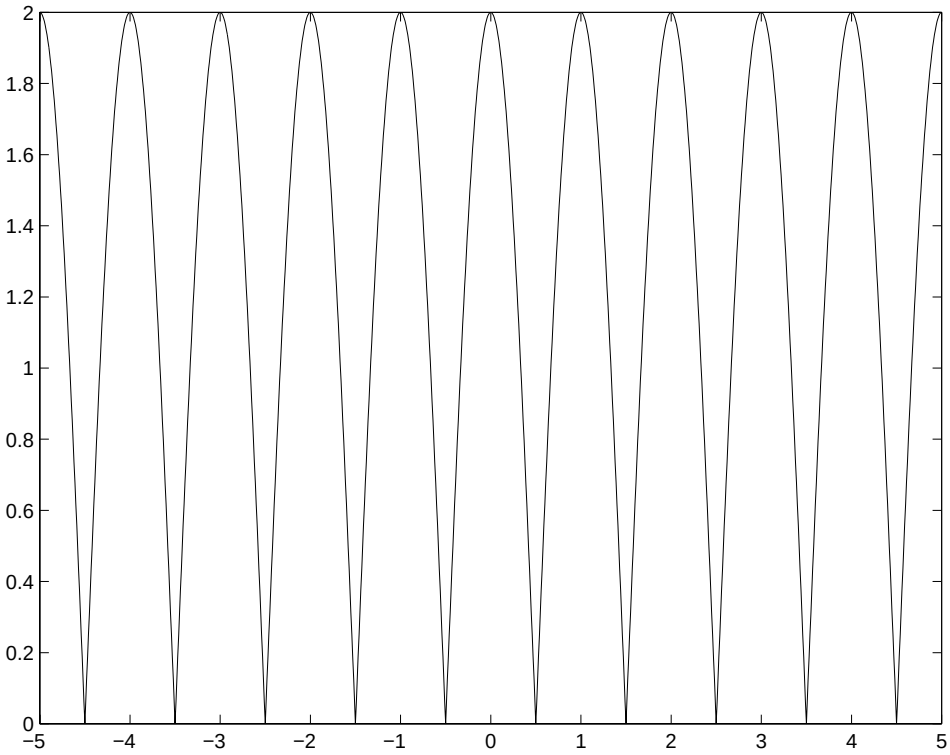
$$\begin{aligned} H(\omega) &= \int_{-\infty}^{\infty} h(t)e^{-i\omega t}dt \\ &= \int_{-\infty}^{\infty} (\delta(t-1)+\delta(t-2))e^{-i\omega t}dt \\ &= e^{-i\omega}+e^{-i2\omega}, \end{aligned}$$

using the sifting rule.

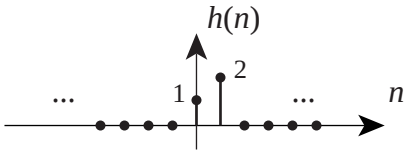
(c) The following Matlab code creates the plot:

```
f = [-5:1/100:5];
H = (exp(-i*2*pi*f)+exp(-i*2*2*pi*f));
plot(f,abs(H));
```

which yields the plot shown in figure below:



4. (a) The impulse response is shown below:



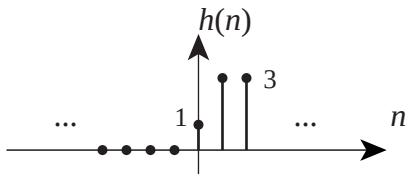
(b) Use convolution to relate the input and output

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{\infty} h(k)x(n-k) \\ &= x(n) + 2x(n-1), \end{aligned}$$

using the sifting rule. When the input is the unit step, this becomes

$$y(n) = u(n) + 2u(n-1) = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n = 0 \\ 3 & \text{if } n \geq 1 \end{cases}$$

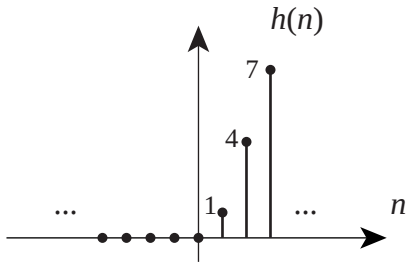
Here is a plot:



(c) If the input is r, then the output is

$$y(n) = r(n) + 2r(n-1) = \begin{cases} 0 & \text{if } n \leq 0 \\ 3n-2 & \text{if } n \geq 1 \end{cases}$$

Here is a plot:



(d) The frequency response is the DTFT of the impulse response,

$$\begin{aligned} H(\omega) &= \sum_{k=-\infty}^{\infty} h(k)e^{-i\omega k} \\ &= 1 + 2e^{-i\omega}. \end{aligned}$$

(e) For all $\omega \in$ *Reals*,

$$\begin{aligned} H(\omega + 2\pi) &= 1 + 2e^{-i(\omega+2\pi)} \\ &= 1 + 2e^{-i\omega}e^{-i2\pi} \\ &= 1 + 2e^{-i\omega}, \text{ since } e^{-i2\pi} = 1 \\ &= H(\omega). \end{aligned}$$

(f) For all $\omega \in$ *Reals*,

$$\begin{aligned} H(-\omega) &= 1 + 2e^{i\omega} \\ &= (1 + 2e^{-i\omega})^* \\ &= H^*(\omega). \end{aligned}$$

(g) The magnitude response is

$$\begin{aligned} |H(\omega)| &= |1 + 2e^{-i\omega}| \\ &= |1 + 2\cos(\omega) - 2i\sin(\omega)| \\ &= \sqrt{(1 + 2\cos(\omega))^2 + (2\sin(\omega))^2} \\ &= \sqrt{1 + 4\cos(\omega) + 4\cos^2(\omega) + 4\sin^2(\omega)} \\ &= \sqrt{5 + 4\cos(\omega)}. \end{aligned}$$

We have used the facts that for real numbers a and b,

$$|a + ib| = \sqrt{a^2 + b^2}$$

and for any $\omega \in$ *Reals*,

$$\cos^2(\omega) + \sin^2(\omega) = 1.$$

(h) The phase response is

$$\begin{aligned} \angle H(\omega) &= \angle(1 + 2e^{-i\omega}) \\ &= \angle(1 + 2\cos(\omega) - 2i\sin(\omega)) \\ &= \tan^{-1}(-2\sin(\omega)/(1 + 2\cos(\omega))) \\ &= -\tan^{-1}(2\sin(\omega)/(1 + 2\cos(\omega))). \end{aligned}$$

We have used the fact that for real numbers a and b,

$$\angle(a + ib) = \tan^{-1}(b/a).$$

(i) The output will be

$$y(n) = |H(\pi/2)|\cos(\pi n/2 + \pi/6 + \angle H(\pi/2)) + |H(\pi)|\sin(\pi n + \pi/3 + \angle H(\pi)).$$

In this case,

$$H(\pi/2) = 1 - 2i$$

and

$$H(\pi) = -1.$$

So

$$|H(\pi/2)| = \sqrt{5}, \quad \angle H(\pi/2) = -\tan^{-1}(2) \approx 1.107$$

and

$$|H(\pi)| = 1, \quad \angle H(\pi) = \pi.$$

Hence,

$$y(n) = \sqrt{5}\cos(\pi n/2 + \pi/6 + 1.107) + \sin(\pi n + \pi/3 + \pi).$$

Since $\sin(\theta + \pi) = -\sin(\theta)$, this simplifies slightly to

$$y(n) = \sqrt{5}\cos(\pi n/2 + \pi/6 + 1.107) - \sin(\pi n + \pi/3).$$

5. The sawtooth signal has period $p = 1$ second, so its fundamental frequency is 2π radians/second, considerably above the passband of the filter. Thus, only the DC term gets through the filter. The DC term is the average over one period, which is 1/2, so the output is

$$y(n) = 1/2.$$

6. (a) Since the system is causal, $h(n) = 0$ for $n < 0$. In addition, h satisfies

$$h(n) = \delta(n) + \delta(n-1) - \alpha h(n-1)$$

(just let the input be an impulse). Thus,

$$\begin{aligned} h(0) &= 1 \\ h(1) &= (1-\alpha) \\ h(2) &= -\alpha(1-\alpha) \\ h(3) &= \alpha^2(1-\alpha) \\ h(4) &= -\alpha^3(1-\alpha) \\ &\cdots \\ h(n) &= (-\alpha)^{n-1}(1-\alpha) \end{aligned}$$

so

$$h(n) = (-\alpha)^{n-1}u(n-1) + (-\alpha)^n u(n),$$

where $u(n)$ is the unit step function.

(b) Although we could calculate the DTFT of the impulse response, it is easier to just let the input be a complex exponential,

$$x(n) = e^{i\omega n}.$$

The output then will be

$$y(n) = H(\omega)e^{i\omega n}.$$

Hence, the following equation must be satisfied,

$$H(\omega)e^{i\omega n} + \alpha H(\omega)e^{i\omega(n-1)} = e^{i\omega n} + e^{i\omega(n-1)}.$$

We can factor out $e^{i\omega n}$ and divide through by it, getting

$$H(\omega)(1 + \alpha e^{-i\omega}) = 1 + e^{-i\omega}.$$

Hence,

$$H(\omega) = \frac{1 + e^{-i\omega}}{1 + \alpha e^{-i\omega}}.$$

(c) The output will be zero if the frequency ω of the sinusoid is such that $H(\omega) = 0$. This occurs if $e^{-i\omega} = -1$, which occurs if $\omega = \pi$. Thus, the following input will yield zero output:

$$x(n) = \cos(\pi n).$$

(d) The following Matlab code works:

```
omega = [0: pi/400: pi];
H = (1 + exp(-i*omega))./(1 + alpha*exp(-i * omega));
plot(omega, abs(H));
```

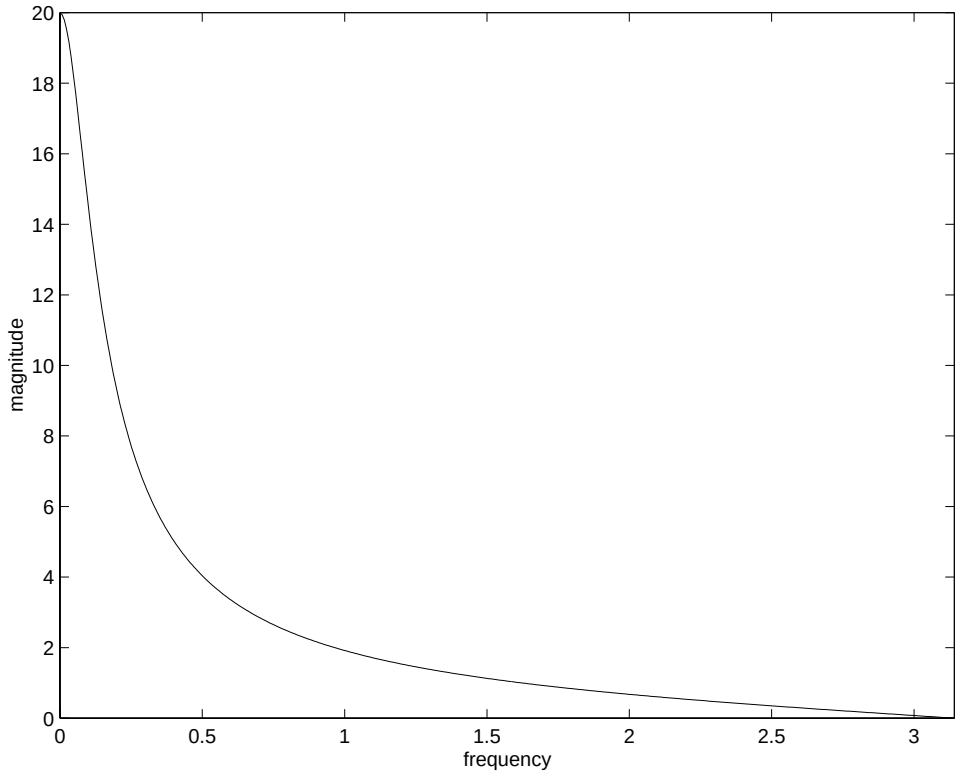


Figure 9.1: A magnitude frequency response.

This results in the plot shown in figure 9.1.

(e) A reasonable choice for the state s is

$$s(n) = [x(n-1), y(n-1)]^T.$$

With this choice,

$$A = \begin{bmatrix} 0 & 0 \\ 1 & \alpha \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ \alpha \end{bmatrix}, \quad d = 1.$$

(f) If $\alpha = 1$, the frequency response becomes $H(\omega) = 1$ and the impulse response becomes $h(n) = \delta(n)$.

7. (a) False. The output frequency may not be the same as the input frequency.

(b) False. You only know the response to one frequency.

(c) True. The frequency response is the DTFT of the impulse response.

(d) True. The impulse response is $y(n) - y(n-1)$, from which you can determine the frequency response.

(e) True. If the system were LTI, the response to the delayed impulse would be the delayed impulse response.

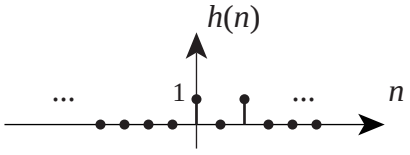
(f) False. The system might be LTI with impulse response given by $h(n) = y(n) - y(n-2) + y(n-4) - y(n-6) + \cdots$.

8. (a) Linear: all except S_3.

(b) Time invariant: S_1, S_2, and S_3.

(c) Causal: S_1, S_3 and S_6.

9. (a) The impulse response is shown below:



(b) Use convolution to relate the input and output

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{\infty} h(k)x(n-k) \\ &= x(n) + x(n-2), \end{aligned}$$

using the sifting rule. When the input is the unit step, this becomes

$$y(n) = u(n) + u(n-2) = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } 0 \le n \le 1 \\ 2 & \text{if } n \ge 2 \end{cases}$$

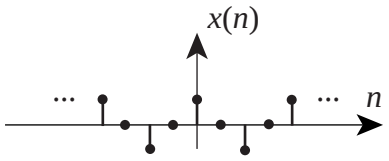
(c) If the input is r, then the output is

$$y(n) = r(n) + r(n-2) = \begin{cases} 0 & \text{if } n < 0 \\ n & \text{if } 0 \le n \le 1 \\ 2n-2 & \text{if } n \ge 2 \end{cases}$$

(d) The output is given by the convolution sum,

$$\begin{aligned} y(n) &= \sum_k h(k)x(n-k) \\ &= x(n) + x(n-2) \end{aligned}$$

where $\sum_k$ means sum over all k. The input looks like this:



Since the output is the sum of two samples separated by one, then by inspection,

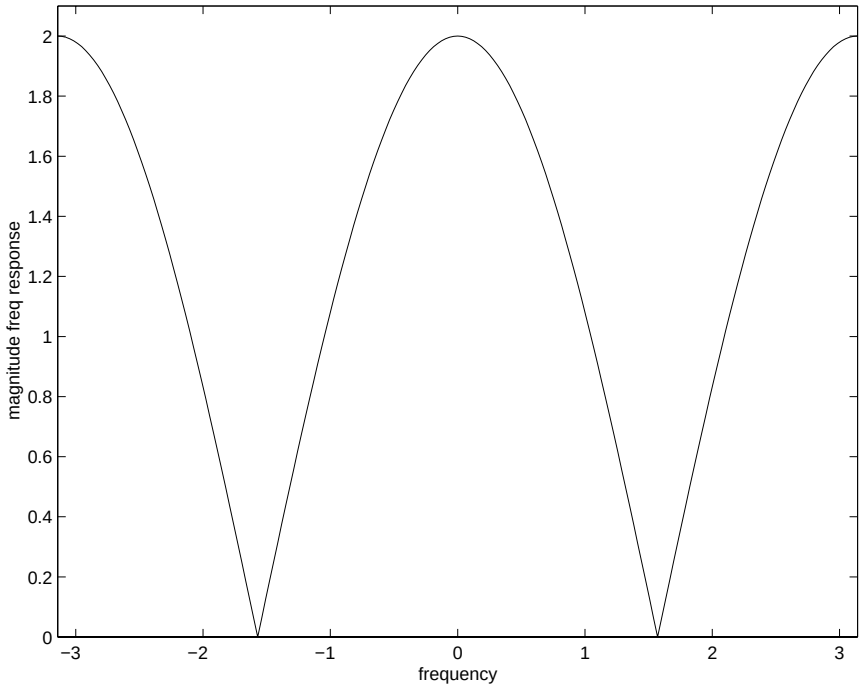
$$y(n) = 0$$

for all n.

(e) The frequency response is

$$\begin{aligned} H(\omega) &= \sum_m h(m)e^{-i2\omega} \\ &= 1 + e^{-i2\omega}. \end{aligned}$$

(f) A plot of the magnitude frequency response is given below:



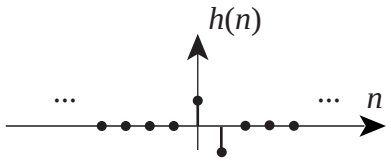
This plot was generated by the following Matlab code:

```
omega = [-pi:pi/100:pi];
H = 1 + exp(-i*2*omega);
plot(omega, abs(H));
```

The frequency response has zero magnitude at frequencies $\pi/2$ and $-\pi/2$, and $\cos(\omega n) = 0.5(e^{i\omega n} + e^{-i\omega n})$, which explains the result in part (b).

(g) The frequency response is periodic with 2π, so a frequency of $\omega' = 2\pi + \pi/2$ will also yield a zero output. However, notice that at this frequency, if $\omega = \pi/2$, then $\cos(\omega' n) = \cos(\omega n)$ for all n, so this input is not really any different.

10. (a) The impulse response is:



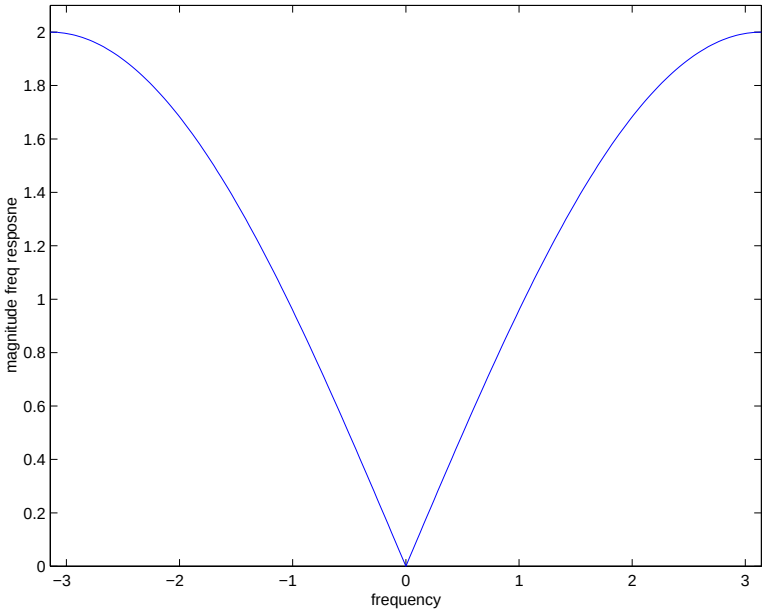
(b) When the input is $x(n) = 1$, the output is

$$\begin{aligned} y(n) &= \textstyle\sum_k h(k)x(n-k) \\ &= x(n) - x(n-1) \\ &= 0. \end{aligned}$$

(c) The frequency response is

$$\begin{aligned} H(\omega) &= \sum_m h(m)e^{-i\omega m} \\ &= 1 - e^{-i\omega}. \end{aligned}$$

(d) A plot of the magnitude frequency response is given below:



This plot was generated by the following Matlab code:

```
omega = [-pi:pi/100:pi];
H = 1 - exp(-i*omega);
plot(omega, abs(H));
```

The frequency response has zero magnitude at zero frequency, which explains the result in part (b).

11. (a) No, since the response to an impulse includes non-zero samples earlier than time zero.

(b) The frequency response is the DTFT of the impulse response,

$$\begin{aligned} H(\omega) &= \sum_{m=-\infty}^{\infty} h(m) e^{-i\omega m} \\ &= \sum_{m=-\infty}^{\infty} (\delta(m-1)/2 + \delta(m+1)/2) e^{-i\omega m} \\ &= (e^{-i\omega} + e^{i\omega})/2 \\ &= \cos(\omega). \end{aligned}$$

This is periodic with period 2π because

$$\forall\, \omega \in \textit{Reals}, \quad \cos(\omega + 2\pi) = \cos(\omega).$$

(c) The fundamental frequency $\omega_0 = \pi/2$, in units of radians per sample. To get the Fourier series coefficients, just write the signal as a sum of complex exponentials,

$$x(n) = (1/2)e^{-i\pi n} + (i/2)e^{-i\pi n/2} + 2 - (i/2)e^{i\pi n/2} + (1/2)e^{-i\pi n},$$

from which we can read off the coefficients,

$$\begin{aligned} X_{-2} &= 1/2 \\ X_{-1} &= i/2 \\ X_0 &= 2 \\ X_1 &= -i/2 \\ X_2 &= 1/2. \end{aligned}$$

The rest of the coefficients are zero.

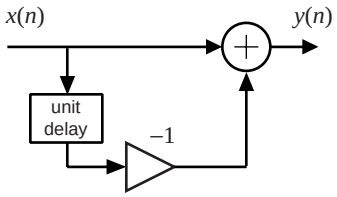
(d) The Fourier series coefficients of the output will be the above Fourier series coefficients multiplied by $H(\omega)$ for the corresponding value of ω. This yields

$$\begin{aligned} y(n) &= -(1/2)e^{-i\pi n} + 2 - (1/2)e^{i\pi n} \\ &= 2 - \cos(\pi n). \end{aligned}$$

12. (a) The step response is

$$y = S(u) = u - D_1(u).$$

(b) The signal flow graph is:



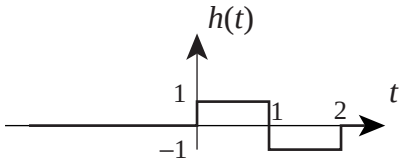
(c) Note that this input x can be written just like the step response

$$x = u - D_1(u).$$

Since the system is LTI, the output is

$$y' = y - D_1(y) = u - 2D_1(u) + D_2(u).$$

This is shown below:



(d) To find the frequency response, let the input be

$$x(t) = e^{i\omega t}.$$

Then the output will be

$$y(t) = H(\omega)e^{i\omega t}.$$

From the signal flow graph, we can also see that the output will be

$$y(t) = e^{i\omega t} - e^{i\omega(t-1)} = (1 - e^{-i\omega})e^{i\omega t}.$$

Thus,

$$H(\omega) = 1 - e^{-i\omega}.$$

13. (a) Use convolution to find the output y when the input is the unit step u,

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{\infty} h(k)u(n-k) \\ &= \sum_{k=0}^{\infty} u(n-k)/2^k \\ &= \begin{cases} \sum_{k=0}^{n} 1/2^k = \frac{1-(1/2)^{n+1}}{1-(1/2)} & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where we have used the suggested identity to simplify the summation.

(b) To find the frequency response, let the input be

$$x(n) = e^{i\omega n}.$$

Then the output is

$$\begin{aligned} H(\omega)e^{i\omega n} &= \sum_{k=0}^{\infty} e^{i\omega(n-k)}/2^k \\ &= e^{i\omega n}\left[\sum_{k=0}^{\infty} e^{-i\omega k}/2^k\right] \end{aligned}$$

Eliminating $e^{i\omega n}$ from both sides we get

$$\begin{aligned} H(\omega) &= \sum_{k=0}^{\infty} e^{-i\omega k}/2^k \\ &= \sum_{k=0}^{\infty} (e^{-i\omega}/2)^k \\ &= \frac{1}{1-e^{-i\omega}/2}. \end{aligned}$$

The magnitude and phase can be plotted using Matlab as follows:

```
omega = [0:pi/500:pi];
H = 1./(1-exp(-i*omega)/2);
subplot(2,1,1);
plot(omega, abs(H));
axis([0,pi,0,2]);
xlabel('frequency (in radians/sec)');
ylabel('magnitude');
subplot(2,1,2);
plot(omega, angle(H));
axis([0,pi,-pi/2,0]);
xlabel('frequency (in radians/sec)');
ylabel('phase');
```

The result is shown in figure 9.2.

(c) This is simply the square of the frequency response of S,

$$H^2(\omega) = \left[\frac{1}{1-e^{-i\omega}/2}\right]^2.$$

(d) Using the results of the previous chapter, the frequency response is

$$\frac{H(\omega)}{1+H(\omega)} = \frac{1}{2-e^{-i\omega}/2}.$$

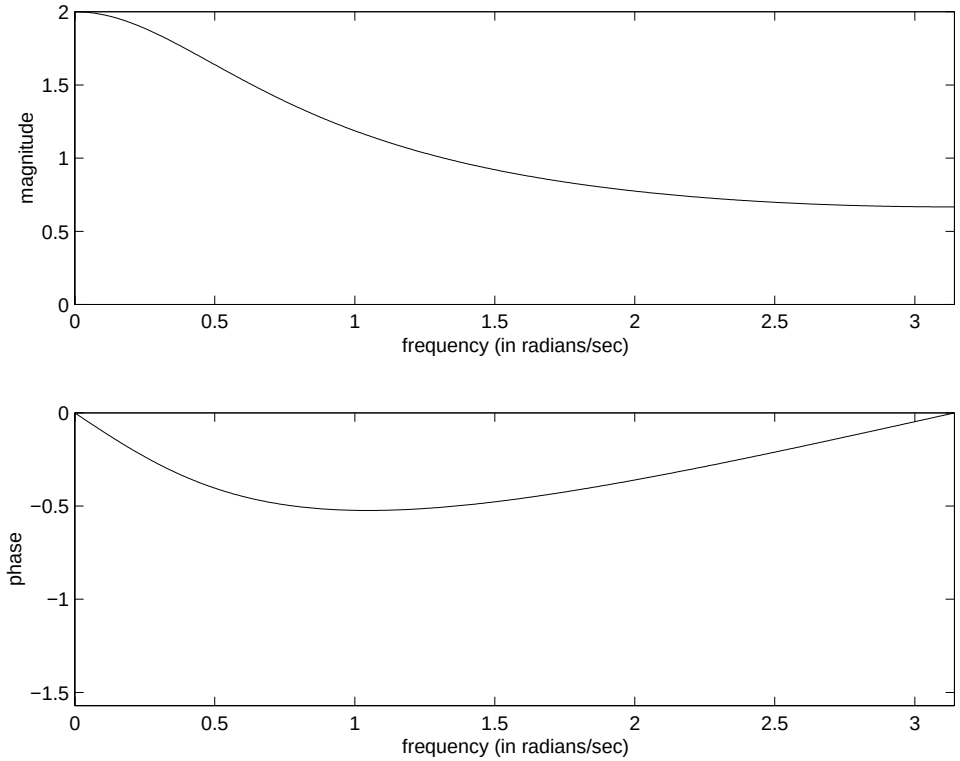


Figure 9.2: Magnitude and phase response of a discrete-time LTI system.

Chapter 10

The Four Fourier Transforms Solutions

1. Let w be the output of the first system when the input is x. Then its DTFT is

$$W(\omega) = H_1(\omega)X(\omega).$$

This becomes the DTFT of the input to the second system, so its output is

$$Y(\omega) = H_2(\omega)W(\omega) = H_2(\omega)H_1(\omega)X(\omega).$$

2. (a) Plugging the specified x into the formula for the Fourier series coefficients yields

$$X_m = \frac{1}{p}\int_0^p e^{-i(m-1)\omega_0 t}dt.$$

When $m = 1$, this becomes

$$X_1 = \frac{1}{p}\int_0^p dt = 1,$$

as desired. When $m \neq 1$,

$$X_m = \frac{1}{p}\int_0^p \cos(-(m-1)\omega_0 t))dt + \frac{i}{p}\int_0^p \sin(-(m-1)\omega_0 t))dt.$$

Since $\omega_0 = 2\pi/p$, each integral is over an integer number of cycles of a sinusoid, which results in an integration to zero. Thus, when $m \neq 1$, $X_m = 0$, as desired.

(b) Using Euler's relation we can write

$$\forall\, t \in \textit{Reals}, \quad ,x(t) = \frac{1}{2}e^{i\omega_0 t} + \frac{1}{2}e^{-i\omega_0 t}.$$

From part (a), we know the Fourier series for $e^{j\omega_0 t}$. We can use this to find the Fourier series for $e^{-i\omega_0 t}$, the complex conjugate, and then use linearity to find the Fourier series for x. The property we will use is that if $y(t) = x^*(t)$ for all t, then $Y_m = X^*_{-m}$ for all m. Thus, if we let $y(t) = e^{-i\omega_0 t}$, then from part (a),

$$\forall\, m \in \textit{Integers}, \quad Y_m = \begin{cases} 1 & \text{if } m = -1 \\ 0 & \text{otherwise} \end{cases}$$

We then use linearity to get the desired result.

(c) Using Euler's relation we can write

$$\forall\, t \in \textit{Reals}, \quad , x(t) = \frac{1}{2i} e^{i\omega_0 t} - \frac{1}{2i} e^{-i\omega_0 t}.$$

Then the proof follows exactly as in part (b).

(d) Plugging the specified x into the formula for the Fourier series coefficients yields

$$X_m = \frac{1}{p} \int_0^p e^{-im\omega_0 t} dt.$$

When $m = 0$, this becomes

$$X_1 = \frac{1}{p} \int_0^p dt = 1,$$

as desired. When $m \neq 0$,

$$X_m = \frac{1}{p} \int_0^p \cos(-m\omega_0 t))dt + \frac{i}{p} \int_0^p \sin(-m\omega_0 t))dt.$$

Since $\omega_0 = 2\pi/p$, each integral is over an integer number of cycles of a sinusoid, which results in an integration to zero. Thus, when $m \neq 0$, $X_m = 0$, as desired. Notice that any (integer) value for p leads to the same conclusion.

3. (a) The Fourier series coefficients are given by

$$\begin{aligned} X_m &= \frac{1}{p} \int_0^p x(t) e^{-im\omega_0 t} dt \\ &= \frac{1}{p} \int_0^T e^{-im\omega_0 t} dt + \frac{1}{p} \int_{p-T}^p e^{-im\omega_0 t} dt. \end{aligned}$$

Notice that when $m = 0$, this integral is easy to evaluate because $e^0 = 1$, so

$$X_0 = 2T/p,$$

as required. Using the suggested integration formula for each of these integrals, this becomes

$$X_m = \frac{i}{m\omega_0 p} \left[e^{-im\omega_0 T} - e^0 + e^{-im\omega_0 p} - e^{-im\omega_0 (p-T)} \right]$$

Notice that

$$\omega_0 = 2\pi/p$$

so

$$\omega_0 p = 2\pi.$$

Hence,

$$e^{-im\omega_0 p} = 1$$

and

$$e^{-im\omega_0 (p-T)} = e^{-im\omega_0 p} e^{im\omega_0 T} = e^{im\omega_0 T}.$$

Since $e^0 = 1$, the coefficients can be written

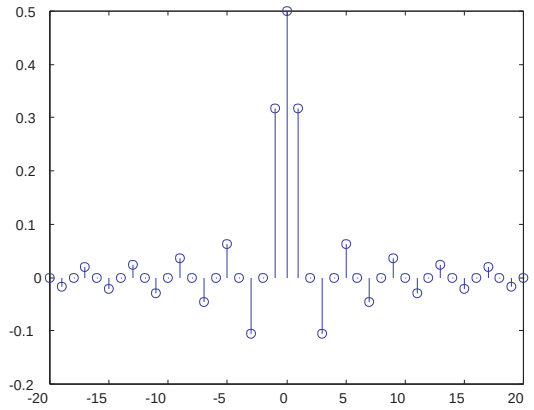
$$\begin{aligned} X_m &= \frac{i}{m\omega_0 p}\left[e^{-im\omega_0 T} - 1 + 1 - e^{im\omega_0 T}\right] \\ &= \frac{i}{m\omega_0 p}\left[-2i\sin(m\omega_0 T)\right] \\ &= \frac{2\sin(m\omega_0 T)}{m\omega_0 p} \\ &= \frac{\sin(m\omega_0 T)}{m\pi,} \end{aligned}$$

as required.

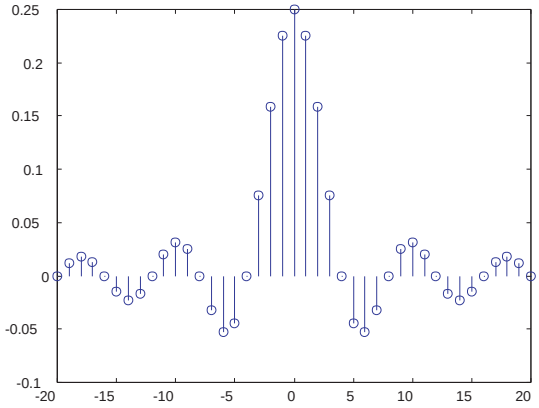
(b) The Fourier series coefficients can be plotted using the following Matlab code:

```
T = 0.5;
p = 2;
m = [-20:1:20];
m1 = [-20:1:-1];
m2 = [1:1:20];
y = [sin(m1*(2*pi/p)*T)./(m1*pi), 2*T/p, sin(m2*(2*pi/p)*T)./(m2*pi)
stem(m,y);
```

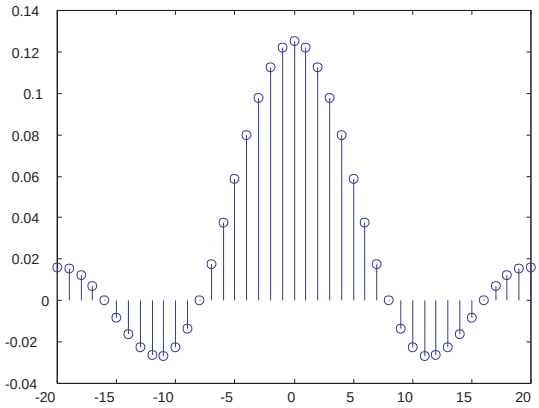
Notice that we have split the range of m into two parts and have separately dealt with $m = 0$ to avoid the "divide by zero" error. The resulting plot is shown below:



Letting $p = 4$ and repeating the above, we get the following plot:



Letting $p = 8$ and repeating the above, we get the following plot:



4. Over the suggested integration interval, there is exactly one impulse, located at zero. Using the sifting rule for Dirac delta functions, and the fact that $e^0 = 1$, the result follows immediately.

5. Let $f = m/p$ so that $2\pi f = m\omega_0$.

$$\begin{aligned} X'_k &= \sum_{n=0}^{p-1} x(n) e^{-ink\omega_0} \\ &= \sum_{n=0}^{p-1} e^{i2\pi fn} e^{-ink\omega_0} = \sum_{n=0}^{p-1} e^{i(m-k)\omega_0} \\ &= \begin{cases} p & \text{if } (m-k) \text{ is a multiple of } p \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

(b) From (a), if $y(n) = e^{-i2\pi fn}$, its DFT coefficients are

$$Y'_k = \begin{cases} p & \text{if } (-m-k) \text{ is a multiple of } p \\ 0 & \text{otherwise} \end{cases}$$

By linearity, the DFT coefficients of $\cos(2\pi fn) = 1/2[e^{i2\pi fn} + e^{-i2\pi fn}]$ are $1/2[X'_k + Y'_k]$.

(c) By linearity the DFT coefficients of $\sin(2\pi fn) = 1/(2i)[e^{i2\pi fn} - e^{-i2\pi fn}]$ are $1/(2i)[X'_k + Y'_k]$.

(d) Since $1 = e^{i2pifn}$ for $f = 0$, the result follows from (a) by taking $m = 0$.

6. (a) We can find its DFT as follows,

$$\begin{aligned} X'_k &= \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0} \\ &= \sum_{m=0}^{M} e^{-imk\omega_0} + \sum_{m=p-M}^{p-1} e^{-imk\omega_0} \\ &= \sum_{m=0}^{M} e^{-imk\omega_0} + e^{iMk\omega_0} \sum_{m=0}^{M-1} e^{-imk\omega_0}, \end{aligned}$$

where the last step is accomplished by changing variables and simplifying using the fact that $\omega_0 = 2\pi/p$. Notice that when $k = 0$, this sum is easy to evaluate because $e^0 = 1$, so

$$X'_0 = 2M + 1,$$

as required. Moreover, when k is any multiple of p, we get $X'_k = 2M + 1$, as it should be, since the DFT is periodic with period p. Now comes some tedium. We combine the above summations as follows,

$$\begin{aligned} X'_k &= e^{-iMk\omega_0} + (1 + e^{iMk\omega_0}) \sum_{m=0}^{M-1} e^{-imk\omega_0} \\ &= e^{-iMk\omega_0} + (1 + e^{iMk\omega_0}) \frac{1 - e^{-iMk\omega_0}}{1 - e^{-ik\omega_0}} \\ &= \frac{e^{iMk\omega_0} - e^{-i(M+1)k\omega_0}}{1 - e^{-ik\omega_0}} \\ &= \frac{e^{-ik0.5\omega_0}(e^{i(M+0.5)k\omega_0} - e^{-i(M+0.5)k\omega_0})}{e^{-ik0.5\omega_0}(e^{ik0.5\omega_0} - e^{-ik0.5\omega_0}} \\ &= \frac{\sin((M + 0.5)k\omega_0)}{\sin(0.5k\omega_0)} \end{aligned}$$

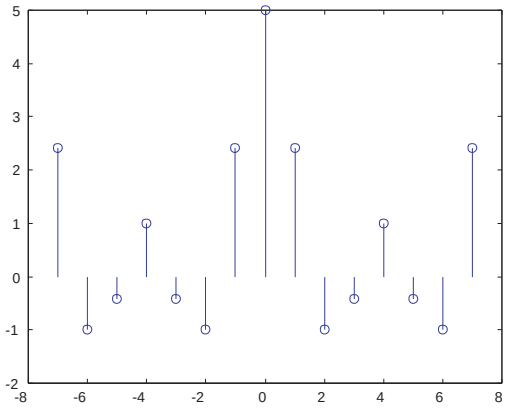
where we have used the suggested identity to get the second line.

(b) The Fourier series coefficients can be plotted using the following Matlab code:

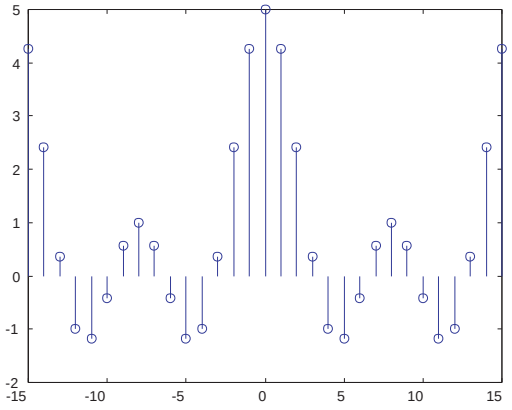
```
M = 2;
p = 32;
k = [-p+1:1:p-1];
k1 = [-p+1:1:-1];
k2 = [1:1:p-1];
y1 = sin(k1*(2*pi/p)*(M+0.5))./sin(k1*0.5*(2*pi/p));
```

```
y2 = sin(k2*(2*pi/p)*(M+0.5))./sin(k2*0.5*(2*pi/p));
y = [y1, 2*M+1, y2];
stem(k,y);
```

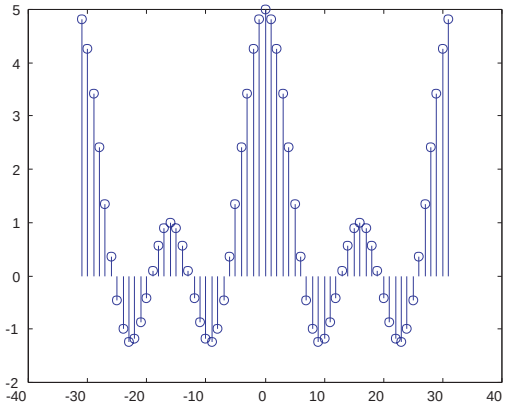
Notice that we have split the range of k into two parts and have separately dealt with $k = 0$ to avoid the "divide by zero" error. The resulting plot is shown below:



Letting $p = 16$ and repeating the above, we get the following plot:



Letting $p = 32$ and repeating the above, we get the following plot:



7. Over the summation interval, there is exactly one impulse, located at zero. Using the sifting rule for Kronecker delta functions, and the fact that $e^0 = 1$, the result follows immediately.

8.

$$\begin{aligned}
X(\omega) &= \sum_{m=-\infty}^{\infty} x(m)e^{-i\omega m} \\
&= \sum_{m=-M}^{M} e^{-i\omega m} \\
&= \sum_{m=0}^{2M} e^{-i\omega(m-M)} \quad \text{by change of variables} \\
&= e^{i\omega M} \sum_{m=0}^{2M} e^{-i\omega m} \\
&= e^{i\omega M} \frac{1 - e^{-i\omega(2M+1)}}{1 - e^{-i\omega}} \quad \text{by summation identity} \\
&= \frac{e^{i\omega M} - e^{-i\omega(M+1)}}{1 - e^{-i\omega}} \\
&= \frac{e^{-i\omega/2}(e^{i\omega(M+1/2)} - e^{-i\omega(M+1/2)})}{e^{-i\omega/2}(e^{i\omega/2} - e^{-i\omega/2})} \\
&= \frac{\sin(\omega(M+1/2))}{\sin(\omega/2)}
\end{aligned}$$

as desired.

9. From the Fourier series table we see that the Fourier series for x is

$$\forall\, m \in \textit{Integers}, \quad X_m = \begin{cases} 1/2i & \text{if } m = 1 \\ -1/2i & \text{if } m = -1 \\ 0 & \text{otherwise} \end{cases}$$

There are only two non-zero Fourier series coefficients. We can write down its Fourier transform using the relation between the Fourier series and the Fourier transform.

10. (a)

$$\forall\, n \in \textit{Integers}, \quad x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} 2\pi\delta(\omega - \omega_0) e^{i\omega n} d\omega.$$

Using the sifting rule, this evaluates to

$$x(n) = e^{i\omega_0 n}.$$

(b) It is periodic with period p, so from the table of DFTs, its DFT is

$$\forall\, k \in \textit{Integers}, \quad X'_k = \begin{cases} p & \text{if } k \in \{\cdots 1-2p, 1-p, 1, 1+p, 1+2p, \cdots\} \\ 0 & \text{otherwise} \end{cases}$$

(c) Again using the sifting rule, we get

$$x(n) = \sum_{k=-\lfloor p/2 \rfloor}^{\lfloor p/2 \rfloor} X_k e^{ik\omega_0}$$

so X_k are the discrete Fourier series coefficients.

(d) Yes, it is periodic with period p. Using the relationship between the DFS and the DFT, we can write

$$X(\omega) = \sum_{k=-\lfloor p/2 \rfloor}^{\lfloor p/2 \rfloor} \frac{X'_k}{p} \delta(\omega - k\omega_0),$$

where X'_k are the DFT coefficients.

11. By the definition of y and the definition of the Fourier transform,

$$Y(\omega) = \int_{-\infty}^{\infty} x(at) e^{-i\omega t} dt.$$

We can change variables on this, letting $\tau = at$, so that $d\tau = a dt$, then

$$Y(\omega) = \frac{1}{|a|} \int_{-\infty}^{\infty} x(\tau) e^{-i\omega\tau/a} d\tau.$$

Comparing this with the definition of the Fourier transform of X we see that

$$Y(\omega) = \frac{1}{|a|} X(\omega/a).$$

12. The DTFT of the signal $z(n) = e^{i\omega_1 n} x(n)$ is

$$\begin{aligned} Z(\omega) &= \sum_{m=-\infty}^{\infty} e^{i\omega_1 n} x(n) e^{-i\omega n} \\ &= \sum_{m=-\infty}^{\infty} e^{-i(\omega-\omega_1)n} = X(\omega - \omega_1) \end{aligned}$$

Because

$$y(n) = \sin(\omega_1 n)x(n) = \frac{1}{2i}[e^{i\omega_1 n}x(n) - e^{-i\omega_1 n}x(n)],$$

by linearity of the DTFT we get,

$$Y(\omega) = \frac{1}{2i}[X(\omega - \omega_1) - X(\omega + \omega_1)].$$

13. (a) We have

$$\begin{aligned} Y_m &= \frac{1}{p}\int_0^p x(t-\tau)e^{-im\omega_0 t}dt \\ &= \frac{1}{p}\int_\tau^{p+\tau} x(s)e^{-im\omega_0 s}e^{-im\omega_0 t}ds \\ &= X(m)e^{-im\omega_o \tau} \end{aligned}$$

(b) We have

$$\begin{aligned} Y_m &= \frac{1}{p}\int_0^p x(t)e^{i\omega_1 t}e^{-im\omega_0 t}dt \\ &= \frac{1}{p}\int_0^p x(t)e^{-\omega_0(m-M)t}dt = X_{m-M} \end{aligned}$$

(c) From $\cos(\omega_1 t) = 1/2[e^{i\omega_1 t} + e^{-i\omega_1 t}]$, part (b), and linearity, we get

$$Y_m = \frac{1}{2}(X_{m-M} + X_{m+M}).$$

(d) From $\sin(\omega_1 t) = 1/(2i)[e^{i\omega_1 t} - e^{-i\omega_1 t}]$, part (b), and linearity, we get

$$Y_m = \frac{1}{2i}(X_{m-M} - X_{m+M}).$$

14. (a) The definition of the DTFT is

$$Y(\omega) = \sum_{m=-\infty}^{\infty} y(m)e^{-i\omega n}.$$

The elements of this summation are zero except when m is a multiple of N, so we can write it as

$$Y(\omega) = \sum_{k=-\infty}^{\infty} y(kN)e^{-i\omega kN}.$$

By the definition of y, this is

$$Y(\omega) = \sum_{k=-\infty}^{\infty} x(k)e^{-i\omega kN}.$$

Comparing with the definition of the DTFT of x, we recognize this as

$$Y(\omega) = X(\omega N).$$

(b) Note that

$$\begin{aligned} W(\omega) &= \sum_{k=-\infty}^{\infty} x(n)e^{i\alpha n}e^{-i\omega n} \\ &= \sum_{k=-\infty}^{\infty} x(n)e^{-i(\omega-\alpha)n} \\ &= X(\omega-\alpha). \end{aligned}$$

(c) Note that from Euler's relation,

$$z(n) = x(n)(e^{i\alpha n} + e^{-i\alpha n})/2.$$

Using part (b) and the linearity of the DTFT, it immediately follows that

$$Z(\omega) = X(\omega-\alpha)/2 + X(\omega+\alpha)/2.$$

15. This structure can be recognized as an FIR filter. Its impulse response is

$$h(n) = b_0\delta(n) + b_1\delta(n-1) + b_2\delta(n-2) + b_3\delta(n-3).$$

The DTFT of this will be the frequency response,

$$\begin{aligned} H(\omega) &= \sum_{m=-\infty}^{\infty} h(m)e^{-i\omega m} \\ &= b_0 + b_1e^{-i\omega} + b_2e^{-i2\omega} + b_3e^{-i3\omega}. \end{aligned}$$

The new system has impulse response

$$h'(n) = b_0\delta(n) + b_1\delta(n-2) + b_2\delta(n-4) + b_3\delta(n-6),$$

so

$$\begin{aligned} H'(\omega) &= \sum_{m=-\infty}^{\infty} h(m)e^{-i\omega m} \\ &= b_0 + b_1e^{-i2\omega} + b_2e^{-i4\omega} + b_3e^{-i6\omega} \\ &= H(2\omega). \end{aligned}$$

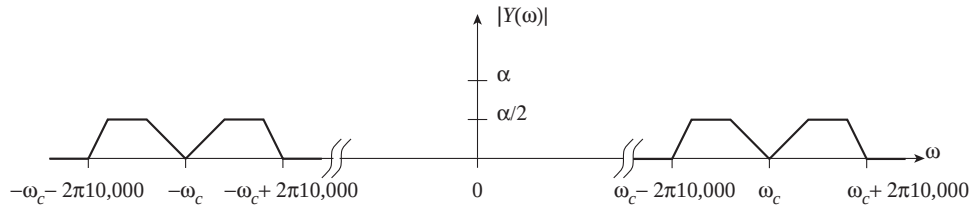
16. (a) Write

$$\begin{aligned} y(t) &= x(t)cos(\omega_c t) \\ &= (x(t)/2)e^{i\omega_c t} + (x(t)/2)e^{-i\omega_c t}. \end{aligned}$$

Thus, plugging this into the definition of the CTFT we get,

$$Y(\omega) = X(\omega-\omega_c)/2 + X(\omega+\omega_c)/2.$$

(b) The original Fourier transform is shifted up and down by ω_c and scaled by half, as shown below:



(c) Note that

$$\begin{aligned} w(t) &= y(t)cos(\omega_c t) \\ &= (y(t)/2)e^{i\omega_c t} + (y(t)/2)e^{-i\omega_c t}. \end{aligned}$$

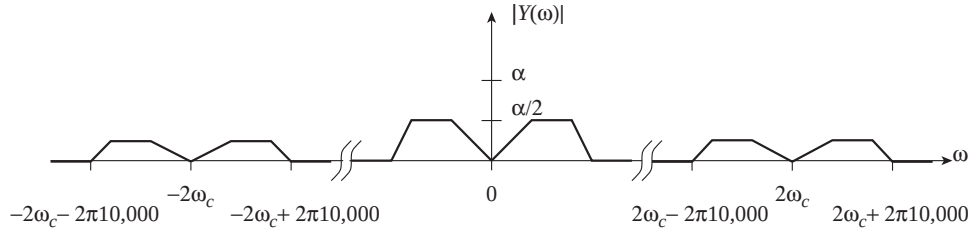
Thus,

$$W(\omega) = Y(\omega - \omega_c)/2 + Y(\omega + \omega_c)/2.$$

Using Y from part (a),

$$W(\omega) = (X(\omega - 2\omega_c)/2 + X(\omega) + X(\omega + 2\omega_c)/2)/2.$$

This is sketched below:



(d) The lowpass filter will eliminate all but the center portion of the figure above, so

$$Z(\omega) = X(\omega)/2$$

or

$$z(t) = x(t)/2.$$

The original audio signal is recovered, albeit attenuated by a factor of 2.

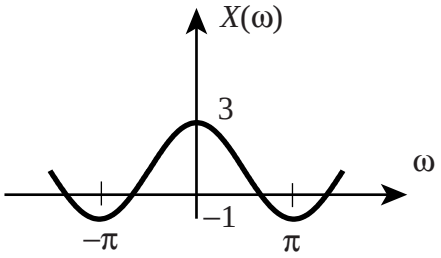
17. (a) The DTFT is

$$\begin{aligned} X(\omega) &= \sum_{m=-\infty}^{\infty} x(m)e^{-i\omega m} \\ &= \sum_{m=-\infty}^{\infty} (\delta(m-1) + \delta(m) + \delta(m+1))e^{-i\omega m} \\ &= e^{-i\omega} + 1 + e^{i\omega} \\ &= 1 + 2p\cos(\omega). \end{aligned}$$

This is periodic with period 2π because

$$\forall\ \omega \in \textit{Reals}, \quad 1 + 2\cos(\omega + 2\pi) = 1 + 2\cos(\omega).$$

A sketch is shown below:



(b) Since the system is LTI,

$$\begin{aligned} Y(\omega) &= H(\omega)X(\omega) \\ &= e^{-i\omega}(e^{i\omega} + 1 + e^{-i\omega}) \\ &= 1 + e^{-i\omega} + e^{-i2\omega}. \end{aligned}$$

(c) $y = \textit{InverseDTFT}(Y)$. Use linearity to avoid evaluating the integral. Let

$$Y_1(\omega) = 1,\ Y_2(\omega) = e^{-i\omega},\ Y_1(\omega) = e^{-i2\omega},$$

each of which is recognizable as the DTFT of a delay,

$$y_1(n) = \delta(n),\ y_2(n) = \delta(n-1),\ y_3(n) = \delta(n-2).$$

Combining these, we get

$$y(n) = \delta(n) + \delta(n-1) + \delta(n-2).$$

(d) The sketches are shown below:

includegraphics../book/fourier/rectDelayedSolution.eps

18. (a) Letting the input be the Kronecker delta function, $x = \delta$, we get that the impulse response must satisfy

$$h(n) = \delta(n) + ah(n-1).$$

Since the system is causal, we know that $h(n) = 0$ for all $n < 0$, so

$$\begin{aligned} h(0) &= 1 \\ h(1) &= a \\ h(2) &= a^2 \\ &\cdots \\ h(n) &= a^n, \quad \forall\ n \geq 0. \end{aligned}$$

We can write this compactly as

$$h(n) = a^n u(n),$$

where $u(n)$ is the unit step function.

(b) Substituting $x(n) = e^{i\omega n}$ and $y(n) = H(\omega)e^{i\omega n}$ into the difference equation, we get

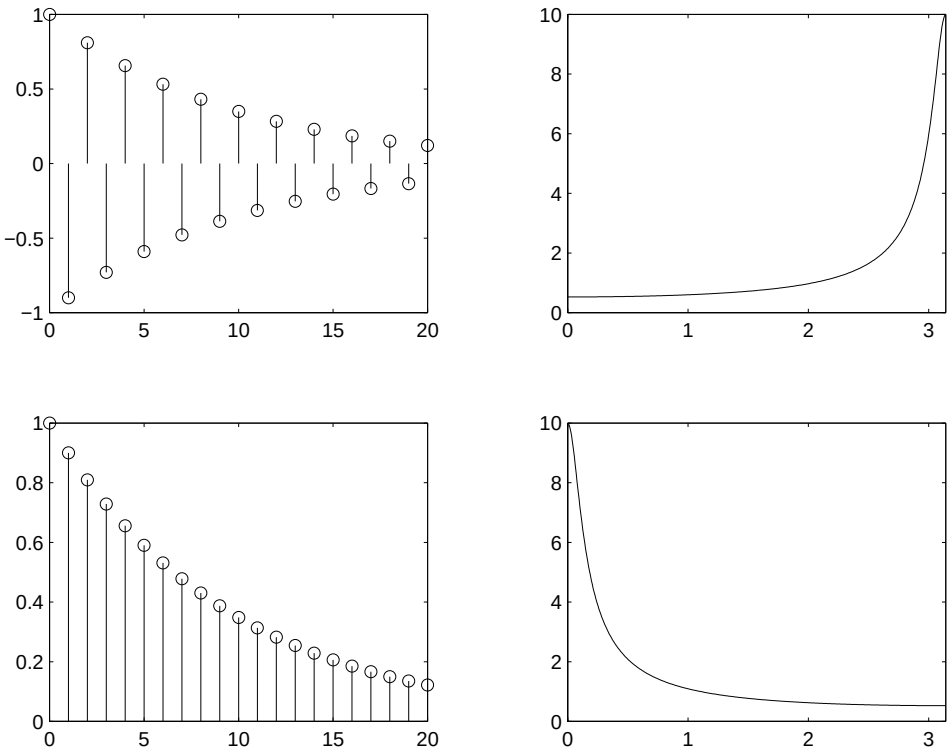
$$H(\omega)e^{i\omega n} = e^{i\omega n} + aH(\omega)e^{i\omega n}e^{i\omega}.$$

Solving for $H(\omega)$ we get

$$H(\omega) = \frac{1}{1 - ae^{i\omega}}.$$

(c) Since the DTFT of the impulse response is the frequency response, the result follows immediately from parts (a) and (b).

(d) The plots are shown below, where the upper plots correspond to $a = -0.9$ and the lower plots to $a = 0.9$:



This was constructed using the following Matlab commands:

```
a = -0.9;
n = 0:20;
h = a.^n;
subplot(2,2,1); stem(n, h);
omega = 0:pi/400:pi;
H = 1./(1 - a*exp(-i*omega));
subplot(2,2,2); plot(omega, abs(H)); axis([0,pi,0,10]);
a = 0.9;
h = a.^n;
subplot(2,2,3); stem(n, h);
H = 1./(1 - a*exp(-i*omega));
```

```
subplot(2,2,4); plot(omega, abs(H)); axis([0,pi,0,10]);
```

19. (a) Note that

$$X(-\omega) = i\sin(-K\omega) = -i\sin(K\omega) = X^*(-\omega),$$

using the fact that $\sin(\theta) = -\sin(-\theta)$. Thus, X is conjugate symmetric, which implies that x is real.

(b) Using Euler's relation,

$$X(\omega) = (e^{iK\omega} - e^{-iK\omega})/2.$$

We can recognize the inverse DTFT of each of these terms to get

$$x(n) = (\delta(n+K) - \delta(n-K))/2$$

where δ is the Kronecker delta function.

20. First, note that y is periodic with period p, just as x is. Its Fourier series coefficients are given by the formula

$$\begin{aligned}
Y_m &= \frac{1}{p}\int_0^p y(t)e^{-im\omega_0 t}dt \\
&= \frac{1}{p}\int_0^p x(t-\tau)e^{-im\omega_0 t}dt \\
&= \frac{1}{p}\int_{-\tau}^{p-\tau} x(t)e^{-im\omega_0 (t+\tau)}dt \\
&= e^{-im\omega_0\tau}\frac{1}{p}\int_{-\tau}^{p-\tau} x(t)e^{-im\omega_0 t}dt \\
&= e^{-im\omega_0\tau}\frac{1}{p}\int_0^p x(t)e^{-im\omega_0 t}dt \\
&= e^{-im\omega_0\tau}X_m,
\end{aligned}$$

where we have changed variables in the integral (replacing t with $t-\tau$), and then changed the limits from $-\tau$ to $p-\tau$ to 0 to p. The change of limits is valid because we are integrating over one cycle of a periodic function, so it does not matter where the integral begins. The end result is

$$Y_m = e^{-im\omega_0\tau}X_m,$$

so just as with a CTFT, a time delay affects Fourier series coefficients by multiplying them by a complex exponential.

21. Use the inverse CTFT,

$$\begin{aligned} x(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{i\omega_0 t}d\omega \\ &= \frac{T}{2\pi}\int_{-\pi/T}^{\pi/T} e^{i\omega_0 t}d\omega \\ &= \frac{T}{2\pi it}[e^{it\pi/T} - e^{-it\pi/T}] \\ &= \frac{\sin(t\pi/T)}{t\pi/T}. \end{aligned}$$

22. Use the CTFT,

$$\begin{aligned} Y(\omega) &= \int_{-\infty}^{\infty} y(t)e^{-i\omega t}dt \\ &= \int_{-\infty}^{\infty} X(t)e^{-i\omega t}dt \end{aligned}$$

so

$$\begin{aligned} \frac{1}{2\pi}Y(-\omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} X(t)e^{i\omega t}dt \\ &= x(\omega), \end{aligned}$$

recognizing this as an inverse CTFT with symbols ω and t swapped. Thus,

$$\frac{1}{2\pi}Y(-\omega) = x(\omega)$$

which implies that

$$Y(\omega) = 2\pi x(-\omega).$$

23. Define

$$y(t) = X(t) = 2\pi\frac{\sin(at)}{at}.$$

From exercise 21, with π/T replaced by a,

$$Y(\omega) = \begin{cases} (2\pi)\pi/a, & \text{if } |\omega| \le a \\ 0, & \text{if } |\omega| > a \end{cases}$$

From exercise 22,

$$Y(\omega) = 2\pi x(-\omega)$$

so

$$x(t) = \frac{1}{2\pi}Y(-t).$$

Hence,

$$x(t) = \begin{cases} \pi/a, & \text{if } |t| \leq a \\ 0, & \text{if } |t| > a \end{cases}$$

Chapter 11

Sampling and Reconstruction Solutions

1. (a) The fundamental frequency is

$$\omega_0 = 10\pi \text{ radians/second}.$$

(b) The Fourier series coefficients are

$$A_0 = 0, A_1 = 1, A_2 = 1, A_3 = 1, A_k = 0 \text{ for } k > 3,$$

and

$$\phi_k = 0 \text{ for all } k.$$

(c) The sampled signal is

$$\begin{aligned} y(n) &= \cos(10\pi n/10) + \cos(20\pi n/10) + \cos(30\pi n/10) \\ &= 1 + 2\cos(\pi n). \end{aligned}$$

The fundamental frequency is therefore $\omega_0 = \pi$ radians/sample.

(d) The DFS coefficients are

$$A_0 = 1, A_1 = 2, \phi_1 = 0.$$

There are no more coefficients, since the period is $p = 2$.

(e) The "smoothest" (lowest frequency content) interpolating signal is

$$w(t) = 1 + 2\cos(10\pi t).$$

(f) Yes, there is aliasing distortion. The 10 Hz cosine has been aliased down to DC, and the 15 Hz cosine has been aliased down to 5 Hz, overlapping the 5 Hz cosine.

(g) Sampling at twice the highest frequency will work. The highest frequency is 15 Hz, so sampling at 30 Hz will avoid aliasing distortion.

2. We need to show that if $y_1 = \textit{Sampler}_T(x_1)$ and $y_2 = \textit{Sampler}_T(x_2)$ then

$$ay_1 + by_2 = \textit{Sampler}_T(ax_1 + bx_2).$$

This follows because $\textit{Sampler}_T(ax_1 + bx_2)$ is given by

$$ax_1(nT) + bx_2(nT)$$

which is equal to

$$ay_1(n) + by_2(n).$$

3. Note that $\cos(\theta) = \cos(-\theta)$. Therefore,

$$\cos(-2\pi 440nT + \phi) = \cos(2\pi 440nT - \phi).$$

Thus, $f = 440$ and $\theta = -\phi$.

4. (a) Yes, it is periodic with period $p = 1$.

(b) Note that

$$y(n) = \sum_{k=-\infty}^{\infty} r(n-k) = 1$$

because only one term in the summation is non-zero, and its value is 1.

(c) For $T = 0.5$, $y = \textit{Sampler}_T(x)$ is given by

$$y(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ 0 & \text{if } n \text{ is odd} \end{cases}$$

For $z = \textit{IdealInterpolator}_T(\textit{Sampler}_T(x))$, we need a signal that contains only frequencies less than or equal to the Nyquist frequency that passes through these samples. The following signal works

$$z(t) = (1/2)(1 + \cos((\pi/T)t)).$$

Note: It is a slightly more subtle fact that this is the *only* signal that works. This fact is hinted at in exercise 7 below.

(d) The signal x is a square wave, having abrupt transitions, and hence components with arbitrarily high frequency. Since the Nyquist-Shannon sampling theorem says we have to sample at twice the highest frequency, and there is no highest frequency, then there is nothing we can do. No T works.

5. (a) The highest frequency in the Fourier series expansion is $4\omega_0 = \pi$ radians per second. The Nyquist-Shannon sampling theorem says that $x = \textit{IdealInterpolator}_T(\textit{Sampler}_T(x))$ if we sample at twice that frequency, or 2π radians/second, or 1 sample/second. Thus, any $T \leq 1$ works.

(b) If $T = 4$, then from part (a) we expect aliasing distortion. We get

$$\begin{aligned} y(n) &= \sum_{k=0}^{4} \cos(k(\pi/4)n4) \\ &= \sum_{k=0}^{4} \cos(k\pi n) \\ &= \cos(0) + \cos(\pi n) + \cos(2\pi n) + \cos(3\pi n) + \cos(4\pi n). \end{aligned}$$

The first, third, and last of these terms are always 1 for any integer n. The second and fourth alternate between $+1$ and -1, depending on whether n is even or odd. Thus,

$$y(n) = \begin{cases} 5 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

This can be written

$$y(n) = 3 + 2\cos(\pi n).$$

Notice that the frequencies at 2π and 4π got aliased down to DC, while the frequency at 3π got aliased down to π.

(c) We need a continuous-time signal that has no frequency terms higher than the Nyquist frequency, yet passes through the samples. Since $T = 4$, the Nyquist frequency is $\pi/4$. By inspection, this is

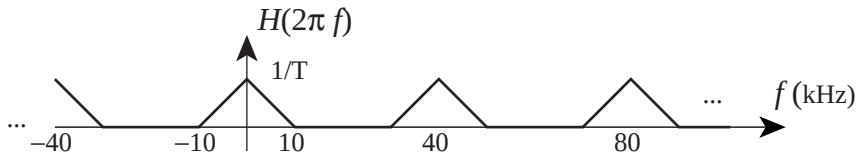
$$w(t) = 3 + 2\cos(\pi t/4).$$

Note that there is a subtlety here. The following answer also works:

$$w(t) = 3 + 2\cos(\pi t/4) + a\sin(\pi t/4)$$

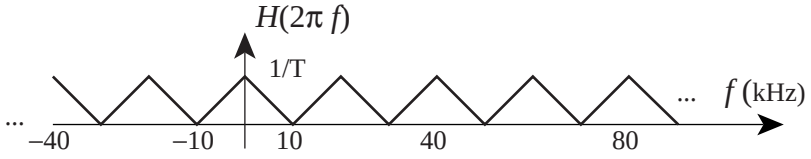
for any constant a because the last term, when sampled with $T = 4$, always yields zero. This ambiguity is a feature of sinusoids at the Nyquist frequency.

6. (a) The sketch is shown below:



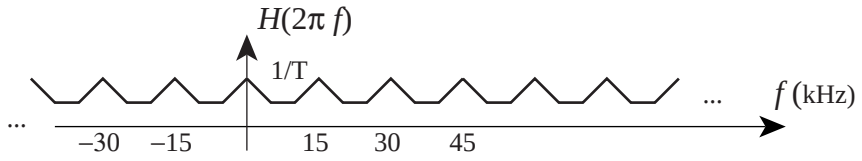
The height of each of the peaks is $1/T$, which in this case is 40,000.

(b) The sketch is shown below:



The height of each of the peaks is $1/T$, which in this case is 20,000.

(c) The sketch is shown below:



The height of each of the peaks is $1/T$, which in this case is 15,000. Notice that the overlapping CTFTs caused aliasing distortion.

7. The sampled sinusoids are

$$y_1(n) = \cos(\omega_1 nT)$$
$$y_2(n) = \cos(\omega_2 nT).$$

Evaluating these at $n = 1$ we get

$$y_1(1) = \cos(\omega_1 T)$$
$$y_2(1) = \cos(\omega_2 T).$$

We will show the required fact by contraposition. That is, we want to show that if assertion A is true then assertion B is true. We do this by showing that if B is false, then A must be false. Suppose then that B is false, i.e. that

$$\textit{Sampler}_T(x_1) = \textit{Sampler}_T(x_2).$$

We need to show that A is false, i.e. that

$$\omega_1 = \omega_2.$$

If $\textit{Sampler}_T(x_1) = \textit{Sampler}_T(x_2)$, then

$$y_1(1) = y_2(1)$$

or

$$\cos(\omega_1 T) = \cos(\omega_2 T).$$

But since $0 \leq \omega_1 \leq \pi/T$ and $0 \leq \omega_2 \leq \pi/T$ and $\omega_1 \neq \omega_2$ this says that

$$\cos(\theta_1) = \cos(\theta_2)$$

for two distinct θ_1, θ_2 in the range $0 \leq \theta_1, \theta_2 \leq \pi$. But the cosine function is one-to-one in this range, so θ_1 and θ_2 cannot be distinct. I.e., it would have to be true that $\theta_1 = \theta_2$, which would imply that $\omega_1 = \omega_2$, contradicting assertion A.

Chapter 12

Stability

1. (a) The Z transform $\hat{X}$ of x is

$$\forall z \in \textit{RoC}(x), \quad \hat{X}(z) = \sum_{m=-\infty}^{\infty} a^m u(-m) z^{-m} = \sum_{m=-\infty}^{0} a^m z^{-m} = \frac{1}{1 - a^{-1} z} \, ,$$

where

$$\textit{RoC}(x) = \{z \in \textit{Complex} \mid \sum_{n=0}^{\infty} |a^{-1} z|^n < \infty\} = \{z \in \textit{Complex} \mid |z| < |a|\}.$$

(b) There is one pole at $z = a$.

(c) The signal is absolutely summable if $|a| > 1$.

(d) The DTFT is

$$\forall\, \omega \in \textit{Reals}, \quad X(\omega) = \hat{X}(e^{i\omega}) == \frac{1}{1 - a^{-1} e^{i\omega}} \, .$$

2. (a) We have $\forall z \in \textit{RoC}(x)$,

$$\hat{X}(z) = \sum_{m=-M}^{M} z^{-m} = z^M \sum_{k=0}^{2M} z^{-k} = z^M \frac{1 - z^{-(2M+1)}}{1 - z^{-1}} = \frac{z^{2M+1} - 1}{z^M (z - 1)} \, ,$$

where

$$\textit{RoC}(x) = \{z \in \textit{Complex} \mid \sum_{m=-M}^{M} |z^{-m}| < \infty\} = \textit{Complex}.$$

(b) The numerator has roots at the $2M + 1$ roots of unity, which are of the form

$$z = e^{i 2K\pi/(2M+1)},$$

for integers K. There are $2M + 1$ distinct complex numbers of this form. All of these are zeros, except the one at $K = 0$, which is $z = 1$, because that one is cancelled by the denominator factor $(z - 1)$. There are M poles at the origin.

(c) The signal is always absolutely summable.

(d) The DTFT is

$$\forall\, \omega \in \textit{Reals}, \quad X(\omega) = \hat{X}(e^{i\omega}) == \frac{e^{i\omega(2M+1)} - 1}{e^{i\omega M}(e^{i\omega} - 1)}\,.$$

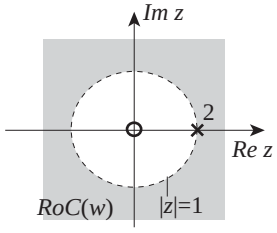
3. (a) The Z transform of the **unit ramp** signal $w(n) = nu(n)$, for all $n \in \textit{Integers}$, is

$$\begin{aligned} \hat{W}(z) &= \sum_{m=0}^{\infty} mz^{-m} = \sum_{m=0}^{\infty} (m+1)z^{-m} - \sum_{m=0}^{\infty} z^{-m} \\ &= \frac{1}{(1-z^{-1})^2} - \frac{1}{1-z^{-1}} = \frac{z^{-1}}{(1-z^{-1})^2}\,, \end{aligned}$$

with region of convergence

$$\textit{RoC}(w) = \{z \in \textit{Complex} \mid |z| > 1\}.$$

(b) The pole-zero plot is:

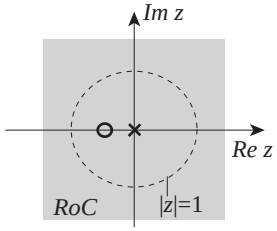


(c) The region of convergence does not include the unit circle, so the signal is not absolutely summable.

4. (a) The Z transform of $h_1(n) = \delta(n) + 0.5\delta(n-1)$ is

$$\hat{H}_1(z) = 1 + 0.5z^{-1} = \frac{z + 0.5}{z}.$$

The pole-zero plot is shown below:

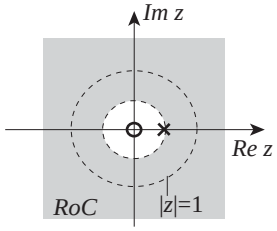


The pole is at $z = 0$ and the zero is at $z = -0.5$. The region of convergence is the entire complex plane except $z = 0$. The system is stable.

(b) The Z transform of $h_2(n) = (0.5)^n u(n)$ is

$$\hat{H}_1(z) = \frac{z}{z - 0.5}.$$

The pole-zero plot is shown below:

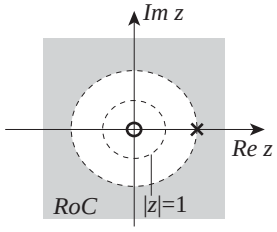


The pole is at $z = 0.5$ and the zero is at $z = 0$. The region of convergence is the shaded area. The system is stable.

(c) The Z transform of $h_3(n) = 2^n u(n)$ is

$$\hat{H}_1(z) = \frac{z}{z-2}.$$

The pole-zero plot is shown below:



The pole is at $z = 2$ and the zero is at $z = 0$. The region of convergence is the shaded area. The system is not stable.

5. (a) The Laplace transform is

$$\begin{aligned} \forall\, s \in \mathit{RoC}(x), \quad \hat{X}(s) &= \int_{-\infty}^{\infty} x(t) e^{-st} dt \\ &= -\int_{-\infty}^{0} e^{-at} e^{-st} dt \\ &= -\int_{-\infty}^{0} e^{-(s+a)t} dt \\ &= \frac{1}{s+a}. \end{aligned}$$

The region of convergence is

$$\mathit{RoC}(x) = \{s \in \mathit{Complex} \mid \mathit{Re}\{s\} < -\mathit{Re}\{a\}\}.$$

(b) There is a pole at $s = -a$, and there are no finite zeros.

(c) x is absolutely integrable if $\mathit{Re}\{a\} < 0$.

(d) The CTFT is the Laplace transform evaluated on the imaginary axis, so

$$X(\omega) = \hat{X}(i\omega) = \frac{1}{i\omega + a}.$$

6. This follows immediately from the definition of the Laplace transform,

$$\begin{aligned}
\hat{W}(s) &= \int_{-\infty}^{\infty} w(t)e^{-st}dt \\
&= \int_{-\infty}^{\infty} (ax(t) + by(t))e^{-st}dt \\
&= a\int_{-\infty}^{\infty} x(t)e^{-st}dt \\
&\quad + b\int_{-\infty}^{\infty} y(t)e^{-st}dt \\
&= a\hat{X}(s) + b\hat{Y}(s).
\end{aligned}$$

The region of convergence of w must include at least the regions of convergence of x and y, since both integrals on the next to last line above must converge. Conceivably, however, the region of convergence may be larger. Thus, all we can assert in general is

$$\mathit{RoC}(w) \supset \mathit{RoC}(x) \cap \mathit{RoC}(y).$$

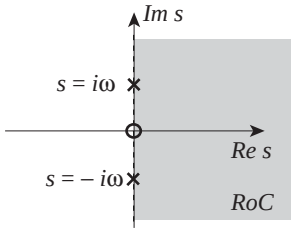
7. (a) Euler's relation implies that

$$y(t) = \frac{1}{2}[e^{i\omega_0 t}u(t) + e^{-i\omega_0 t}u(t)].$$

Using example 12.18 and linearity,

$$\begin{aligned}
\hat{Y}(s) &= \frac{1}{2}\left\{\frac{1}{s + i\omega_0} + \frac{1}{s - i\omega_0}\right\} \\
&= \frac{s}{s^2 + \omega_0^2}\,.
\end{aligned}$$

(b) This has a zero at $s = 0$ and two poles, one at $s = i\omega_0$ and the other at $s = -i\omega_0$. Both of these poles lie on the imaginary axis, as shown below:



The region of convergence is the right half of the complex plane. Note that if this were the impulse response of an LTI system, that system would not be stable. The region of convergence does not include the imaginary axis.

8. This system is stable for $|a| < 1$, since

$$\sum_{m=-\infty}^{\infty} |h(m)| = \sum_{m=0}^{\infty} |a^n \cos(\omega_0 n)| \leq \sum_{m=0}^{\infty} |a^n| = \frac{1}{1-|a|} < \infty,$$

where we have used the fact that $|a^n| = |a|^n$.

9. (a) The Laplace transform of the unit ramp is, using integration by parts,

$$\begin{aligned} \hat{Y}(s) &= \int_0^{\infty} te^{-st} dt \\ &= -t\frac{1}{s}\, e^{-st} \,|_{t=0}^{\infty} + \frac{1}{s}\int_0^{\infty} e^{-st} dt \\ &= -\frac{1}{s^2} e^{-st} \,|_{t=0}^{\infty} = \frac{1}{s^2}\,. \end{aligned}$$

$\lim_{t\to\infty} \frac{t}{s} e^{-st} = 0$ and the integral converges for $Re\{s\} > 0$, so $RoC(y) = \{s \mid Re\{s\} > 0\}$.

(b) The pole-zero plot has two poles at $s = 0$ and no zeros.

10. If these are discrete-time systems,

$$\sum_{-\infty}^{\infty} |ah(n) + bg(n)| \leq |a| \sum_{-\infty}^{\infty} |h(n)| + |b| \sum_{-\infty}^{\infty} |g(n)| < \infty,$$

because h and g are absolutely summable. If these are continuous-time systems,

$$\int_{-\infty}^{\infty} |ah(n) + bg(n)| \leq |a| \int_{-\infty}^{\infty} |h(n)| + |b| \int_{-\infty}^{\infty} |g(n)| < \infty,$$

because h and g are absolutely integrable.

11. Suppose the input is bounded by M, i.e. $|x(n)| \leq M$ for all n (or $|x(t)| \leq M$ for all t). Since the first system is stable, its output v is bounded by some number K. Since the second system is stable (and *its* input v is bounded), its output y is also bounded. Hence the series composition is BIBO stable.

12. (a) $\forall n, y(n) = \sum_{-\infty}^{\infty} h(m)x(n-m)$, so

$$|y(n)| \leq \sum_{-\infty}^{\infty} |h(m)||x(n-m)| \leq M \sum_{-\infty}^{\infty} |h(m)| = M\|h\|.$$

(b) For this input, the output at time 0 is

$$y(0) = \sum_{m=-\infty}^{\infty} h(m)x(-m) = \sum_{-\infty}^{\infty} \frac{h(m)h(m)}{|h(m)|} = \|h\|.$$

Since this input is bounded by 1, by the first part y is bounded by $\|h\|$. Since $y(0) = \|h\|$, this is the smallest bound.

(c) This is immediate from

$$\sum_n |h(n)+g(n)| \le \sum_n |h(n)| + \sum_n |g(n)| = \|h\| + \|g\|.$$

(d) This follows from

$$\begin{aligned}\|h*g\| = \sum_{n=-\infty}^{\infty} |\sum_{m=-\infty}^{\infty} h(n-m)g(m)| &\le \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} |h(n-m)||g(m)| \\ &= \sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} |h(l)||g(m)| = \|h\|\|g\|\end{aligned}$$

13. From exercise 11 we know that the series compositions with impulse responses $h_1 * g_1$ and $h_2 * g_2$ are both stable. From exercise 10 we know that the parallel composition with impulse response $h = h_1 * g_1 + h_2 * g_2$ is stable. From exercise 12 we get a bound on the norm,

$$\|h\| \le \|h_1\|g_1\| + \|h_2\|\|g_2\|.$$

14. The Z transform of x is

$$\hat{X}(z) = \sum_{n=M}^{N} x(n)z^{-n}.$$

If $N > 0$, the sum is finite for $z \neq 0$, so *RoC*$(x) = \{z \mid |z| > 0\}$. If $N \le 0$, the sum if finite for all z, so *RoC*(x) = *Complex*.

(a) If $N > 0$ and $x_N \neq 0$, $\hat{X}(z) = x(M)z^{-M} + \cdots + x(N)z^{-N}$ has N poles at $z = 0$. If $N \le 0$, $\hat{X}(z)$ is a polynomial in z with no finite poles.

(b) This follows from the first part since $N > 0$.

15. (a) By definition of the Laplace transform

$$\begin{aligned}\hat{Y}(s) &= \int_{-\infty}^{\infty} y(t)e^{-st}dt \\ &= \int_{-\infty}^{\infty} \sum_n x(n)\delta(t-nT)e^{-st}dt = \sum_n x(n)e^{-snT}\end{aligned}$$

using the sifting property. Since $\int_{-\infty}^{\infty} \delta(t-nT)dt = 1$, the integral above converges absolutely for all s with

$$\sum_n |x(n)||e^{-snT}| = \sum_n |x(n)||e^{sT}|^{-n} < \infty,$$

which holds if and only if the complex number $e^{sT} \in$ *RoC*(x). So

$$\textit{RoC}(y) = \{s \in \textit{Complex} \mid e^{sT} \in \textit{RoC}(x)\}.$$

(b) $\hat{Y}(s) = \sum_n x(n)e^{-snT} = \hat{X}(e^{sT})$.

(c) From the previous parts,

$$\hat{Y}(s) = \frac{1}{e^{sT}-1},$$

and *RoC*$(y) = \{s \mid e^{sT} \in \textit{RoC}(x)\} = \{s \mid |e^{sT}| > 1\} = \{s \mid \textit{Re}\{s\} > 0\}$.

Chapter 13

Laplace and Z Transforms

1. (a) By Euler's relation,
$$x(n) = \frac{1}{2i}[e^{i\omega_0 n}u(n) - e^{-i\omega_0 n}u(n)].$$
Using the Z transform of the exponential and the linearity of the Z transform,
$$\begin{aligned}\hat{Y}(z) &= \frac{1}{2i}\left\{\frac{z}{z-e^{i\omega_0}} - \frac{z}{z-e^{-i\omega_0}}\right\} \\ &= \frac{1}{2i}\frac{(e^{i\omega_0}-e^{-i\omega_0})z}{(z-e^{i\omega_0})(z-e^{-i\omega_0})} \\ &= \frac{z\sin(\omega_0)}{z^2 - 2z\cos(\omega_0)+1}\,.\end{aligned}$$
where
$$\mathit{RoC}(x) = \{z \in \mathit{Complex} \mid |z| > |e^{i\omega_0}| = 1\}.$$
(b) There is one zero at $z = 0$, and two poles, one at $z = e^{j\omega_0}$ and the other at $z = e^{-i\omega_0}$.
(c) This signal is not absolutely summable.

2. (a) Write this as the sum of two signals,
$$\forall\, n \in \mathit{Integers}, \quad x(n) = x_1(n) + x_2(n),$$
where
$$\forall\, n \in \mathit{Integers}, \quad x_1(n) = a^{|n|}u(n) = a^n u(n),$$
and
$$\forall\, n \in \mathit{Integers}, \quad x_2(n) = a^{|n|}u(-n-1) = a^{-n}u(-n-1).$$
The first of these has Z transform, from the table,
$$\forall\, z \in \{z \in \mathit{Complex} \mid |z| > |a|\}, \quad \hat{X}_1(z) = \frac{z}{z-a}.$$
The second of these can be converted into a signal that is the table by time reversal and delay,
$$\forall\, n \in \mathit{Integers}, \quad x_3(n) = x_2(-n) = a^n u(n-1),$$

and

$$\forall\, n \in \textit{Integers}, \quad x_4(n) = x_3(n+1) = a(a^n u(n)).$$

From the table (and linearity),

$$\forall\, z \in \{z \in \textit{Complex} \mid |z| > |a|\}, \quad \hat{X}_4(z) = \frac{az}{z-a}.$$

From the delay property,

$$\forall\, z \in \{z \in \textit{Complex} \mid |z| > |a|\}, \quad \hat{X}_3(z) = z^{-1}\frac{az}{z-a} = \frac{a}{z-a}.$$

From the time-reversal property,

$$\forall\, z \in \{z \in \textit{Complex} \mid |z^{-1}| > |a|\}, \quad \hat{X}_2(z) = \hat{X}_3(z^{-1}) = \frac{a}{z^{-1}-a} = \frac{az}{1-az}.$$

The latter region of convergence can be written

$$\textit{RoC}(x_2) = \{z \in \textit{Complex} \mid |z| < |a^{-1}|\}.$$

The region of convergence of x is at least,

$$\textit{RoC}(x) \supset \textit{RoC}(x_1) \cap \textit{RoC}(x_2) = \{z \in \textit{Complex} \mid |a| < |z| < 1/|a|\}.$$

This set is non-empty if $|a| < 1$. The Z tranform is therefore,

$$\forall\, z \in \textit{RoC}(x), \quad \hat{X}_1(z) = \frac{z}{z-a} + \frac{a}{z(1-az)} = \frac{-az^3 + z^2 + az - a^2}{z(z-a)(1-az)}\,.$$

(b) There is a pole at $z = 0$, another at $z = a$, and a third at $z = 1/a$.

(c) The region of convergence includes the unit circle if $|a| < 1$, in which case the signal will be absolutely summable.

3. The Z tranform of the input is

$$\forall\, z \in \{z \in \textit{Complex} \mid z \neq 0\}, \quad \hat{X}(z) = 1 - 0.9z^{-1} = \frac{z-0.9}{z}.$$

Multiplying by the transfer function, we get

$$\forall\, z \in \textit{RoC}(y), \quad \hat{Y}(z) = 1.$$

This converges everywhere, so $\textit{RoC}(y) = \textit{Complex}$. Note that we can recognize the output as

$$\forall\, n \in \textit{Integers}, \quad y(n) = \delta(n).$$

4. (a) The Z transform of the sinusoid $x(n) = \cos(\omega_0 n)u(n)$, for all n, is

$$\hat{X}(z) = \frac{z^2 - z\cos(\omega_0)}{z^2 - 2z\cos(\omega_0) + 1}\,,$$

with domain $\textit{RoC}(x) = \{z \in \textit{Complex} \mid |z| > 1\}$. So the Z transform of $y(n) = a^{-n}\cos(\omega_0 n)u(n)$, for all n, is

$$\hat{Y}(z) = \frac{a^2z^2 - az\cos(\omega_0)}{a^2z^2 - 2az\cos(\omega_0) + 1}\,,$$

with domain $\textit{RoC}(y) = \{z \in \textit{Complex} \mid |az| > 1\}$.

(b) $\hat{X}$ has poles at $e^{i\omega_0}, e^{-i\omega_0}$. $\hat{Y}$ has poles at $ae^{i\omega_0}, ae^{-i\omega_0}$.

(c) The signal is absolutely summable if the region of convergence includes the unit circle, which will occur if $|a| > 1$.

5. Fix $r, r_1 < r < r_2$. Pick $\epsilon > 0$ so that $r_1 < r/(1+\epsilon)$ and $r(1+\epsilon) < r_2$. Pick $0 < N < \infty$ so that $n(1+\epsilon)^{-n} < 1$ for $n > N$. (Then we also have $|n(1+\epsilon)^n| < 1$ for $n < -N$.) Now

$$\sum_{-\infty}^{\infty} |nx(n)r^{-n}| = \sum_{-\infty}^{N} |nx(n)r^{-n}| + \sum_{-N+1}^{N-1} |nx(n)r^{-n}| + \sum_{N}^{\infty} |nx(n)r^{-n}|.$$

The first sum on the right can be bounded since

$$\begin{aligned}
\sum_{-\infty}^{N} |nx(n)r^{-n}| &= \sum_{-\infty}^{N} |n(1+\epsilon)^n x(n)[(1+\epsilon)r]|^{-n} \\
&\leq \sum_{-\infty}^{N} |x(n)[(1+\epsilon)r]|^{-n} \leq \sum_{\infty}^{N} |x(n)r_2^{-n}| \\
&\leq \sum_{-\infty}^{\infty} |x(n)r_2^{-n}| < \infty.
\end{aligned}$$

The second sum is bounded since it has a finite number of summands. The third sum can be bounded since

$$\begin{aligned}
\sum_{N}^{\infty} |nx(n)r^{-n}| &= \sum_{N}^{\infty} |n(1+\epsilon)^{-n} x(n)[\frac{r}{1+\epsilon}]^{-n} \\
&\leq \sum_{N}^{\infty} |x(n)r_1^{-n}| \\
&\leq \sum_{-\infty}^{\infty} |x(n)r_1^{-n}| < \infty.
\end{aligned}$$

6. (a) Taking Z transforms on both sides,

$$\forall\, z \in \mathit{RoC}(x) \cap \mathit{RoC}(y), \quad \hat{Y}(z) + b_1 z^{-1}\hat{Y}(z) + b_2 z^{-2}\hat{Y}(z) = a_0\hat{X}(z) + a_1 z^{-1}\hat{X}(z) + a_2 z^{-2}\hat{X}(z).$$

Solving for the transfer function,

$$\forall\, z \in \mathit{RoC}(x) \cap \mathit{RoC}(y), \quad \hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} = \frac{a_0 z^2 + a_1 z^1 + a_2}{z^2 + b_1 z^1 + b_2}\,.$$

(b) The system is causal, so the region of convergence will be the region outside the circle passing through the largest root of the denominator polynomial, $z^2 + b_1 z^1 + b_2$.

(c) The system will be stable if both roots of $z^2 + b_1 z^1 + b_2$ have magnitude smaller than one.

7. Applying the definition of the Laplace transform,

$$\begin{aligned}\hat{Y}(s) &= \int_{-\infty}^{\infty} x(t-\tau)e^{-st}dt \\ &= \int_{-\infty}^{\infty} x(t')e^{-s(t'+\tau)}dt' \\ &= e^{-s\tau}\int_{-\infty}^{\infty} x(t')e^{-st'}dt' \\ &= e^{-s\tau}\hat{X}(s),\end{aligned}$$

where we have performed a change of variables, $t' = t - \tau$. The region of convergence of y is obviously the same as that of x because $e^{-s\tau}$ is finite for any finite s.

8. Combining the convolution with the definition of the Laplace transform,

$$\begin{aligned}\hat{Y}(s) &= \int_{-\infty}^{\infty} y(t)e^{-st}dt \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)h(t-\tau)e^{-st}dtd\tau \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(\tau)h(t')e^{-s(t'+\tau)}dt'd\tau \\ &= \left(\int_{-\infty}^{\infty} x(\tau)e^{-s\tau}d\tau\right)\left(\int_{-\infty}^{\infty} h(t')e^{-st'}dt'\right) \\ &= \hat{X}(s)\hat{H}(s),\end{aligned}$$

where we have used a change of variables, $t' = t + \tau$. The Laplace transform of y converges absolutely at least at values of s where both $\hat{X}$ and $\hat{H}$ converge abolutely, as is evident from the second to last form above, the product of two integrals. Thus,

$$\mathit{RoC}(y) \supset \mathit{RoC}(x) \cap \mathit{RoC}(h).$$

9. (a) This follows because

$$\hat{Y}(s) = \int_{-\infty}^{\infty} x^*(t)(e^{-st}dt = \left[\int_{-\infty}^{\infty} x(t)e^{-s^*t}\right]^* dt = [\hat{X}(s^*)]^*,$$

because

$$(e^{-st})^* = (e^{-(\sigma+i\omega)t})^* = (e^{-(\sigma-i\omega)t}) = e^{-s^*t}$$

(b) If x is real then $\forall\ t\mathit{Reals},\ x(t) = x^*(t)$, so $\hat{X}(s) = [\hat{X}(s^*)]^*$. Hence,

$$0 = \hat{X}(q^*) = (\hat{X}(q))^*.$$

10. Applying the definition of the Laplace transform,

$$\begin{aligned}\hat{Y}(s) &= \int_{-\infty}^{\infty} x(ct)e^{-st}dt \\ &= \frac{1}{|c|}\int_{-\infty}^{\infty} x(\tau)e^{-\frac{s}{c}\,\tau}d\tau \\ &= \frac{1}{|c|}\hat{X}(\frac{s}{c}).\end{aligned}$$

(The magnitude $|c|$ in the second step takes care of the change of variables, $\tau = ct$, for both positive and negative c.) The integral converges absolutely for all s for which $\frac{s}{c} \in \mathit{Roc}(x)$.

11. Applying the definition of the Laplace transform,

$$\begin{aligned}\hat{Y}(s) &= \int_{-\infty}^{\infty} e^{at}x(t)e^{-st}dt = \int_{-\infty}^{\infty} x(t)e^{-(s-a)t}dt \\ &= \hat{X}(s-a).\end{aligned}$$

The integral converges absolutely when $(s-a) \in \mathit{RoC}(x)$.

12. The output has Z transform

$$\hat{Y}(\omega) = \hat{H}(\omega)\hat{X}(\omega) = \frac{z^2 - (\cos\omega_0)z}{(z-e^{i\omega_0})(z-e^{-i\omega_0})}\frac{z}{(z-e^{i\omega_0})},$$

which has a pole at $e^{i\omega_0}$ with multiplicity two. Hence, $y(k)$ includes the unbounded term $ke^{i\omega_0 k}$.

13. (a) From the Z transforms table, x is the causal signal

$$\forall n, \quad x(n) = (n-1)(n-2)3^{n-3}u(n-3).$$

(b) From the Z transforms table again, x is the anti-causal signal

$$\forall n, \quad x(n) = -(2-n)(1-n)3^{n-3}u(-n).$$

A Matlab plot of the two signals is shown in figure 13.1.

14. (a) $\frac{z+2}{(z+1)(z+3)} = \frac{1/2}{z+1} + \frac{1/2}{z+3}$

(b) $\frac{(z+2)^2}{(z+1)(z+3)} = 1 + \frac{1/2}{z+1} + \frac{-1/2}{z+3}$

(c) $\frac{z+2}{z^2+4} = \frac{z+2}{(z+i2)(z-i2)} = \frac{(1+i)/2}{z+2i} + \frac{(1-i)/2}{z-2i}$.

15. Since

$$\frac{(z+2)^2}{(z+1)(z+3)} = 1 + \frac{1/2}{z+1} + \frac{-1/2}{z+3}$$

has a pole at $z = -1$ and $z = -3$, there are three possible *Roc*.

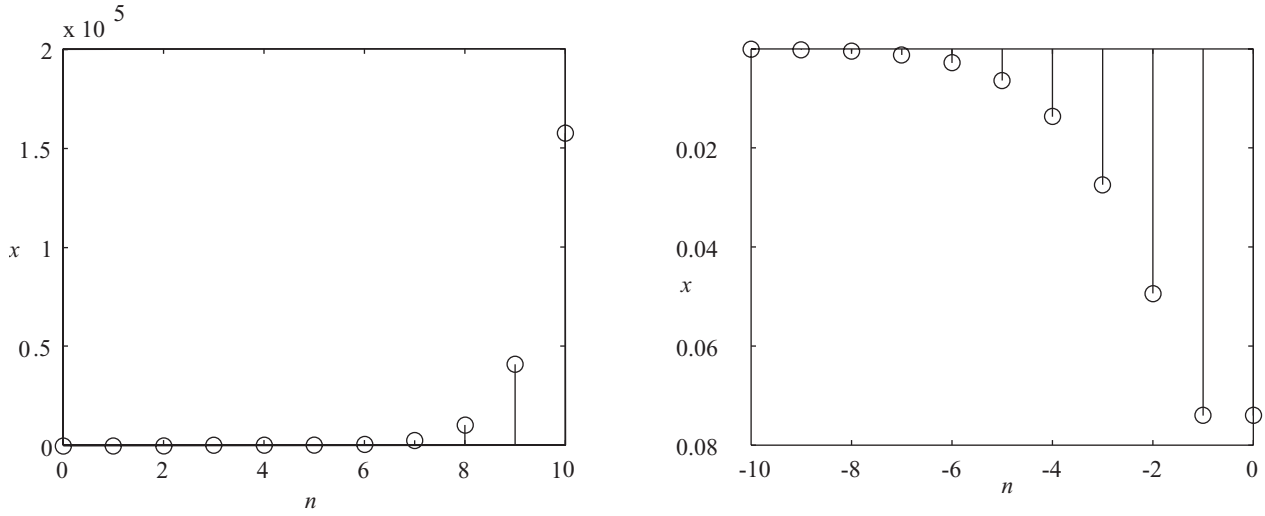


Figure 13.1: Plots for signals in exercise 13. Part (a) on left, part (b) on right.

Case 1. *RoC*$(x) = \{z \mid |z| < 1\}$. x is anti-causal,

$$\forall n, \quad x(n) = \delta(n) + \frac{1}{2}(-1)^n u(-n) + \frac{1}{2}(-3)^{n-1}u(-n).$$

Case 2. *Roc*$(x) = \{z \mid 1 < |z| < 3\}$. x is the two-signal

$$\forall n, \quad x(n) = \delta(n) + \frac{1}{2}(-1)^{n-1} u(n-1) + \frac{1}{2}(-3)^n u(-n).$$

Case 3. *RoC*$(x) = \{z \mid |z| > 3\}$. x is the causal signal

$$\forall n, \quad x(n) = \delta(n) + \frac{1}{2}(-1)^{n-1} u(n-1) - \frac{1}{2}(-3)^{n-1} u(n-1).$$

16. Since

$$\frac{z+2}{z^2+4} = \frac{z+2}{(z+i2)(z-i2)} = \frac{(1+i)/2}{z+2i} - \frac{(1-i)/2}{z-2i},$$

the two possible *RoC* are $\{z \mid |z| < 2\}$ and $\{z \mid |z| > 2\}$.

Case 1. *RoC*$(x) = \{z \mid |z| < 2\}$. So x is anti-causal,

$$\forall n, \quad x(n) = \frac{1+i}{2}(-1)(-2i)^n u(-n) + \frac{1-i}{2}(-1)(2i)^n u(-n).$$

Case 2. *RoC*$(x) = \{z \mid |z| > 2\}$. So x is causal,

$$\forall n, \quad x(n) = \frac{1+i}{2}(-2i)^{n-1} u(n-1) + \frac{1-i}{2}(2i)^{n-1} u(n-1).$$

17. From the Z transforms table, $\hat{H}(z) = z/(z-0.5)$. The Z transform of the unit step is $\hat{X}(z) = z/(z-1)$. Hence the Z transform of the output is

$$\begin{aligned}\hat{Y}(z) &= \hat{H}(z)\hat{X}(z) = \frac{z}{z-0.5} + \frac{z}{z-1} \\ &= 1 + \frac{2}{z-1} - \frac{0.5}{z-0.5}.\end{aligned}$$

RoC$(y) = \{z \mid |z| > 1\}$. So

$$\forall n, \quad \delta(n) + 2u(n-1) - 0.5^n u(n-1).$$

The steady state response is $y_{ss}(n) = 2u(n)$ for all n.

18. From the Z transforms table, denoting Z transforms by upper case letters,

$$\begin{aligned}\hat{U}(z) &= \frac{z}{z-1}\,, \textit{RoC}(u) = \{z \mid |z| > 1\}, \\ \hat{H}(z) &= \frac{1}{1-0.5z}\,, \textit{RoC}(h) = \{z \mid |z| < 2\}, \\ \hat{G}(z) &= \frac{z}{z-0.5}\,, \textit{RoC}(g) = \{z \mid |z| > 0.5\}.\end{aligned}$$

$h * u$ has Z transform

$$\hat{H}(z)\hat{U}(z) = \frac{z}{(1-0.5z)(z-1)} = \frac{2}{1-0.5z} + \frac{2}{z-1}\,,$$

with *RoC*$(h * u) = \{z \mid 1 < z < 2\}$. So $h * u$ is the two-sided signal

$$\forall n, \quad (h * u)(n) = 2^{n+1}u(-n) + 2u(n-1).$$

$g * u$ has Z transform

$$\hat{G}(z)\hat{U}(z) = \frac{z^2}{(z-0.5)(z-1)} = 1 - \frac{0.5}{z-0.5} + \frac{2}{z-1}\,,$$

with *RoC*$(g * u) = \{z \mid |z| > 1\}$. So $g * u$ is the causal signal

$$\forall n, \quad (g * u)(n) = \delta(n) - 0.5^n u(n-1) + 2u(n-1).$$

19. (a)

$$\hat{X}(z) = \frac{z}{z-1}, \quad \hat{Y}(z) = \frac{z}{z-1} - \frac{z}{z-0.5} = \frac{0.5z}{(z-1)(z-0.5)},$$

so its transfer function is

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \frac{0.5}{z-0.5},$$

with region of convergence $\{z \in \textit{Complex} \mid |z| > 0.5\}$.

(b) The system is stable and its frequency response is

$$\forall \omega \in \textit{Reals}, \quad H(\omega) = \frac{0.5}{e^{i\omega} - 0.5}.$$

(c) The impulse response is the inverse Z transform of $\hat{H}$,

$$\forall k \in \textit{Integers}, \quad h(k) = 0.5(0.5)^{k-1}u(k-1) = 0.5^k u(k-1).$$

20. (a)

$$\hat{H}(z) = \frac{1}{1-2z^{-1}}\,,$$

with $\textit{RoC}(h)\{z \in \textit{Complex} \mid |z| > 2\}$.

(b) The input x is the unit step, $x = u$. Then the output y is given by the convolution $y = h * x = h * u$, and for $z \in \textit{RoC}(h) \cap \textit{RoC}(x) = \{z \in \textit{Complex} \mid |z| > 2\}$,

$$\begin{aligned}
\hat{Y}(z) &= \hat{H}(z)\hat{X}(z) \\
&= \frac{1}{1-2z^{-1}}\frac{1}{1-z^{-1}}\,, \\
&= \frac{z^2}{(z-2)(z-1)} \\
&= 1 + \frac{3z-2}{(z-2)(z-1)} \\
&= 1 + \frac{4}{z-2} + \frac{-1}{z-1} \text{ by partial fraction expansion.}
\end{aligned}$$

The region of convergence is $\textit{RoC}(h) \cap \textit{RoC}(x) = \{z \in \textit{Complex} \mid |z| > 2\}$.

(c) Taking inverse Z transform, we get the step response

$$\forall n \in \textit{Integers}, \quad y(n) = \delta(n) + 4 \times 2^{n-1}u(n-1) - u(n-1).$$

21. We take Z Transforms as in section 13.7. The zero-input response is $y_{zi}(n) = 0$ since the initial conditions are 0. So $\hat{Y}(z) = \hat{Y}_{zs}$. In both cases $\hat{X}(z) = z/(z-1)$.

(a) We have

$$\hat{Y}(z) + z^{-2}\hat{Y}(z) = \hat{X}(z),$$

so

$$\begin{aligned}
\hat{Y}(z) &= \frac{z}{(1+z^{-2})}\frac{z}{z-1} = \frac{z^3}{(z^2+1)(z-1)} \\
&= z\{\frac{1/2z}{z^2+1} + \frac{1/2}{z^2+1} + \frac{1/2}{z-1} \\
&\leftrightarrow [\frac{1}{2}\cos(\frac{\pi}{2}n) + \frac{1}{2}\sin(\frac{\pi}{2}n) + \frac{1}{2}]u(n) = y(n).
\end{aligned}$$

(b) We have

$$\hat{Y}(z) + 2z^{-1}\hat{Y}(z) + z^{-2}\hat{Y}(z) = \hat{X}(z),$$

so

$$\begin{aligned}\hat{Y}(z) &= \frac{\hat{X}(z)}{1 + 2z^{-1} + z^{-2}} = \frac{z^3}{(z=1)^2(z-1)} \\ &= z\{\frac{3/4z}{(z+1)^2} - \frac{1/2}{(z+1)^2} + \frac{1/4}{z-1} \\ &\leftrightarrow \frac{3}{4}(n+1)(-1)^n u(n) - \frac{1}{2}n(-1)^{n-1}u(n-1) + \frac{1}{2}u(n) = y(n).\end{aligned}$$

22. We take Laplace Transforms as in section 13.7.1.

(a) We have

$$5(s\hat{Y}(s) - \bar{y}(0)) + 10\hat{Y}(s) = 2\hat{X}(s),$$

so

$$\hat{Y}(s) = \frac{\bar{y}(0)}{s+2} + \frac{2}{5s+10}\hat{X}(s),$$

where $\bar{y}(0) = 2$, $\hat{X}(s) = 1/s$.
The zero-input response is given by

$$\hat{Y}_{zi}(s) = \frac{2}{s+2} \leftrightarrow 2e^{-2t}u(t) = y_{zi}(t);$$

the zero-state response is given by

$$\hat{Y}_{zs}(s) = \frac{2}{5(s+10)s} = \frac{1}{5s} - \frac{1}{5(s+2)} \leftrightarrow [\frac{1}{5} - \frac{1}{5}e^{-2t}]u(t) = y_{zs}(t).$$

(b) We have

$$s^2\hat{Y}(s) - s\bar{y}(0) - \bar{y}^{(1)}(0) + 5(s\hat{Y}(s) - \bar{y}(0)) + 6\hat{Y}(s) = -4\hat{X}(s) - 3s\hat{X}(s),$$

so

$$\hat{Y}(s) = \frac{(5+s)\bar{y}(0) + \bar{y}^{(1)}(0)}{s^2+5s+6} - \frac{3s+4}{s^2+5s+6}\hat{X}(s),$$

where $\bar{y}(0) = -1, \bar{y}^{(1)}(0) = 5, \hat{X}(s) = 1/(s+1)$.
The zero-input response is given by

$$\hat{Y}_{zi}(s) = \frac{-s}{s^2+5s+6} = \frac{2}{s+2} - \frac{3}{s+3} \leftrightarrow [2e^{-2t} - 3e^{-3t}]u(t) = y_{zi}(t);$$

the zero-state response is given by

$$\begin{aligned}\hat{Y}_{zs} &= -\frac{3s+4}{(s^2+5s+6)(s+1)} = \frac{-1/2}{s+1} + \frac{-2}{s+2} + \frac{5/2}{s+3} \\ &\leftrightarrow [-\frac{1}{2}e^{-t} - 2e^{-2t} + \frac{5}{2}e^{-3t}]u(t) = y_{zs}(t).\end{aligned}$$

(c) We have

$$s^2\hat{Y}(s) - s\bar{y}(0) - \bar{y}^{(1)}(0) + 4\hat{Y}(s) = 8\hat{X}(s),$$

so

$$\hat{Y}(s) = \frac{s\bar{y}(0)) + \bar{y}^{(1)}(0)}{s^2+4} + \frac{8}{s^2+4}\hat{X}(s),$$

where $\bar{y}(0) = 1, \bar{y}^{(1)}(0) = 2, \hat{X}(s) = 1/s$.
The zero-input response is given by

$$\hat{Y}_{zi}(s) = \frac{s}{s^2+4} + \frac{2}{s^2+4} \leftrightarrow [\cos 2t + \sin 2t]u(t) = y_{zi}(t);$$

the zero-state response is given by

$$\hat{Y}_{zs}(s) = \frac{8}{s(s^2+4)} = \frac{2}{s} + \frac{-2s}{s^2+4} \leftrightarrow [2 - 2\cos 2t]u(t).$$

(d) We have

$$s^2\hat{Y}(s) - s\bar{y}(0) - \bar{y}^{(1)}(0) + 2(s\hat{Y}(s) - \bar{y}(0)) + 5\hat{Y}(s) = s\hat{X}(s),$$

so

$$\hat{Y}(s) = \frac{(s+2)\bar{y}(0) + \bar{y}^{(1)}(0)}{s^2+2s+5} + \frac{s}{s^2+2s+5}\hat{X}(s),$$

where $\bar{y}(0) = 2, \bar{y}^{(1)} = 0, \hat{X}(s) = 1/(s+1)$.
The zero-input response is given by

$$\begin{aligned} \hat{Y}_{zi}(s) &= \frac{2s+4}{s^2+2s+5} = \frac{2(s+1)}{(s+1)^2+2^2} + \frac{2}{(s+1)^2+2^2} \\ &\leftrightarrow [2e^{-t}\cos 2t + e^{-t}\sin 2t]u(t) = y_{zi}(t). \end{aligned}$$

the zero-state response is given by

$$\begin{aligned} \hat{Y}_{zs}(s) &= \frac{s}{(s+1)((s+1)^2+2^2)} = \frac{-1/4}{s+1} + \frac{s/4+5/4}{(s+1)^2+2^2} \\ &\leftrightarrow [-\frac{1}{4}e^{-t} + \frac{1}{4}e^{-t}\cos 2t + \frac{1}{2}e^{-t}\sin 2t]u(t) = y_{zs}(t). \end{aligned}$$

23. From $y(n) - 2y(n-1) - 3y(n-2) = x(n)$, $s_1(n) = y(n-1), s_2(n) = ay(n-2)$, we get

$$\begin{aligned} s_1(n+1) &= y(n) = 2y(n-1) + 3y(n-2) + x(n) = 2s_1(n) + \frac{3}{a}s_2(n) + x(n) \\ s_2(n+1) &= ay(n-1) = as_1(n) \\ y(n) &= 2s_1(n) + \frac{3}{a}s_2(n) + x(n) \end{aligned}$$

from which we can read off A, b, c, d and check that they are correct.

24. (a) The current through the capacitor is

$$I = C\dot{v}.$$

The KVL equations for the outer loop and the second inner loop are

$$\begin{aligned} u &= R(i+I) + rC\dot{v} + v \\ L\dot{i} &= rI + v \end{aligned}$$

From these three equations we get

$$\begin{aligned} \dot{v} &= -\frac{1}{(R+r)C}v - \frac{R}{(R+r)C}i + \frac{1}{(R+r)C}u \\ \dot{i} &= -\frac{r}{(R+r)L}v - \frac{rR}{(R+r)L}i + \frac{r}{(R+r)L}u \end{aligned}$$

which gives the required differential equation in the form

$$\dot{x}(t) = Ax(t) + bu(t),$$

where

$$A = \begin{bmatrix} -\frac{1}{(R+r)C} & -\frac{R}{(R+r)C} \\ -\frac{r}{(R+r)L} & -\frac{rR}{(R+r)L} \end{bmatrix}, \quad b = \begin{bmatrix} \frac{1}{(R+r)C} \\ \frac{r}{(R+r)L} \end{bmatrix}$$

The c and d in the representation depend on which signal is taken to be the output. For instance, if the capacitor voltage is taken as output, $c^T = [1,\ 0], d = 0$.

(b) The approximation gives the discrete-time system

$$\frac{1}{T}(x_{k+1} - x_k) = Ax_k + bu_k,$$

which is in the form

$$x_{k+1} = Fx_k + gu_k,$$

with

$$F = (I + TA), \quad g = Tb.$$

25. From the example we know the partial fraction expansion of $[sI - A]^{-1}$:

$$[sI - A]^{-1} = \begin{bmatrix} \frac{(3/4)}{s-1} + \frac{(1/4)}{s-5} & \frac{(-1/4)}{s-1} + \frac{(1/4)}{s-5} \\ \frac{(-3/4)}{s-1} + \frac{(3/4)}{s-5} & \frac{(1/4)}{s-1} + \frac{(3/4)}{s-5} \end{bmatrix}.$$

Taking the inverse Laplace transforms yields

$$e^{tA}u(t) = \begin{bmatrix} 3/4 & -1/4 \\ -3/4 & 1/4 \end{bmatrix} e^t u(t) + \begin{bmatrix} 1/4 & 1/4 \\ 3/4 & 3/4 \end{bmatrix} e^{5t} u(t).$$

26. From the example we can get the partial fraction expansion of

$$z[zI - A]^{-1} = \begin{bmatrix} \frac{1/2z}{z-1+2i} + \frac{1/2z}{z-1-2i} & \frac{i/2z}{z-1+2i} + \frac{-i/2z}{z-1-2i} \\ \frac{-i/2z}{z-1+2i} + \frac{i/2z}{z-1-2i} & \frac{1/2z}{z-1+2i} + \frac{1/2z}{z-1-2i} \end{bmatrix}.$$

We can now obtain the inverse Z transform: for all n

$$A^n u(n) = \begin{bmatrix} Re\{(1-2i)^n\} & -Im\{(1-2i)^n\} \\ Im\{(1-2i)^n\} & Re\{(1-2i)^n\} \end{bmatrix}$$

27. (a) The characteristic polynomial of A is $\det[sI - A] = (s-a)^2 + b^2$. So the eigenvalues are $s = a \pm ib$.

(b) The system is stable if and only if both eigenvalues have negative real parts, i.e. $a < 0$.

(c) The Laplace transform of $e^{tA}u(t)$ is

$$[sI - A]^{-1} = \frac{1}{s^2 + b^2} \begin{bmatrix} s-a & -b \\ b & s-a \end{bmatrix}$$

From Laplace transform tables,

$$\begin{aligned} \frac{s}{s^2+b^2} &\leftrightarrow \cos btu(t), \text{ so } \frac{s-a}{(s-a)^2+b^2} \leftrightarrow e^{at}\cos btu(t) \\ \frac{b}{s^2+b^2} &\leftrightarrow \sin btu(t), \text{ so } \frac{b}{(s-a)^2+b^2} \leftrightarrow e^{at}\sin btu(t) \end{aligned}$$

So

$$e^{tA}u(t) = e^{at}\begin{bmatrix} \cos bt & -\sin bt \\ \sin bt & \cos bt \end{bmatrix} u(t).$$

(d) The transfer function is

$$\hat{H}(s) = c^T + [sI - A]^{-1}b + d = \frac{s-a}{(s-a)^2 + b^2}.$$

28. The first matrix

$$A = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}$$

has eigenvalues 1 and 5. To get an eigenvector corresponding to 1 we must find $e = [e_1\ e_2]^T$ such that $Ae = e$. This gives $e = [1\ -1]^T$. To get an eigenvector corresponding to 5 we must find e such that $Ae = 5e$. This gives $e = [1\ 3]^T$.

The second matrix

$$A = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$$

has eigenvalues $1 + 2i$ and $1 - 2i$. To get an eigenvector corresponding to $(1 + 2i)$ we must find $e = [e_1\ e_2]^T$ such that $Ae = (1 + 2i)e$. This gives $e = [1\ i]^T$. To get an eigenvector corresponding to the (complex cojugate) eigenvalue $(1 - 2i)$, we take the complex conjugate eigenvector $e = [1\ -i]^T$.

29. Since $Ae = pe$, $A^2e = A(Ae) = p^2e$, and so on $\cdots$. So $A^ne = p^ne$. It follows that

$$e^{tA}e = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} e = \sum_{k=0}^{\infty} \frac{t^k p^k}{k!} e = e^{pt}e.$$

(a) $s(n) = A^n e$

(b) $s(t) = e^{pt}e$.

30. From (13.49) we get

$$sI - A = \begin{bmatrix} s & -1 & 0 & \cdots & 0 \\ 0 & s & -1 & \cdots & 0 \\ \cdots & \cdots & & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -1 \\ a_0 & a_1 & a_2 & \cdots & a_{N-1} + s \end{bmatrix}.$$

It is now straightforward to check that $[sI - A]^{-1}b$ is indeed as claimed.

Chapter 14

Composition and Feedback Control

1. In all cases, the Z transform of $y - x$ is $\hat{H}_1\hat{H}_2 - 1$.
   (a) $y - x \leftrightarrow \frac{z-0.5}{z-0.5-\epsilon} - 1 = \frac{\epsilon}{z-0.5-\epsilon} \leftrightarrow \epsilon(0.5+\epsilon)^{n-1}u(n-1)$.
   (b) $y - x \leftrightarrow \frac{z-2}{z-2-\epsilon} - 1 = \frac{\epsilon}{z-2-\epsilon} \leftrightarrow \epsilon(2+\epsilon)^{n-1}u(n-1)$.
   (c) In the previous case $|y(n) - x(n)| \to \infty$ if $\epsilon \neq 0, 2 + \epsilon > 1$.

2. (a) Let the error be $d = y - x$. Taking Laplace transforms,

$$\begin{aligned}\hat{D}(s) &= [\hat{H}_1(s)\hat{H}_2(s) - 1]\hat{X}(s) \\ &= [\frac{s+1}{s+2+\epsilon}\,\frac{s+2}{s+1} - 1]\frac{1}{s} \\ &= [\frac{s+2}{s+2+\epsilon} - 1]\frac{1}{s} \\ &= \frac{\frac{\epsilon}{2+\epsilon}}{s+2+\epsilon} + \frac{\frac{-\epsilon}{2+\epsilon}}{s}\end{aligned}$$

   The inverse Laplace transform gives,

$$\forall t, \quad d(t) = \frac{\epsilon}{2+\epsilon}e^{-(2+\epsilon)t}u(t) - \frac{\epsilon}{2+\epsilon}u(t).$$

   The steady state error is $-\frac{\epsilon}{2+\epsilon}$. Figure 14.1 is a Matlab plot for $\epsilon = \pm 0.1$.

   (b) Now

$$\begin{aligned}\hat{D}(s) &= [\frac{s-1}{s+2}\,\frac{s+2}{s-1-\epsilon} - 1]\frac{1}{s} \\ &= [\frac{s-1}{s-1-\epsilon} - 1]\frac{1}{s} = \frac{\epsilon}{1+\epsilon}[\frac{1}{s-1-\epsilon} - \frac{1}{s}]\end{aligned}$$

   The inverse Laplace transform gives,

$$\forall t, \quad d(t) = \frac{\epsilon}{1+\epsilon}\,e^{(1+\epsilon)t}u(t) - \frac{\epsilon}{1+\epsilon}\,u(t).$$

   The first term grows without bound if $1 + \epsilon > 0$.

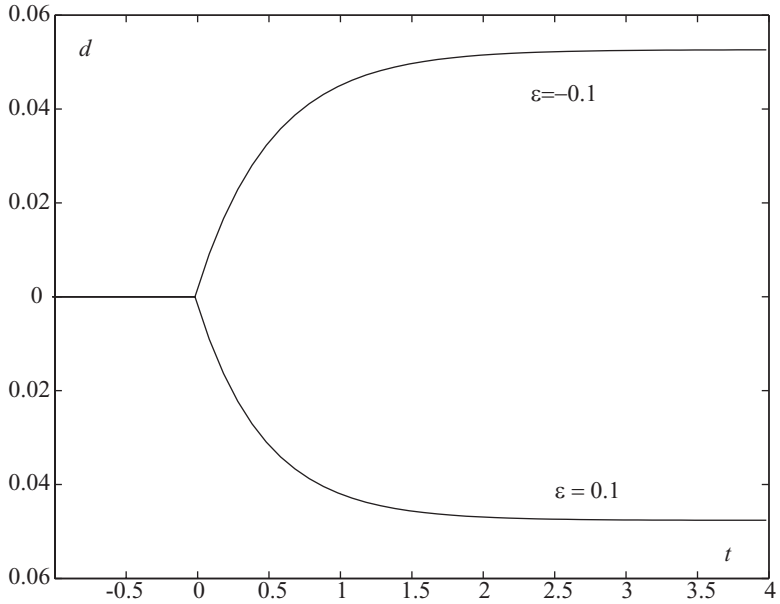


Figure 14.1: Plots for exercise 2.

3. (a) Initially $\hat{H}_1(z) = \frac{z}{z-0.2}$ and $\hat{H}_2(z) = \frac{z-0.2}{z}$. Since both systems are causal, table 13.1 yields

$$\forall n, \quad h_1(n) = (0.2)^n u(n) \quad \text{and } h_2(n) = \delta(n) - 0.2\delta(n).$$

Since $x(n) = 0, n < 0$, we have $v(n) = 0, n < 0$ and for $n \geq 0$,

$$\begin{aligned} v(n) &= \sum_{k=0}^{n} h_1(k)x(n-k) \\ &= x(n) + 0.2x(n-1) + 0.04x(n-2) + 0.008x(n-3) + \cdots \\ &= \begin{cases} 1, & n = 0 \\ 0.2, & n = 1 \\ 1.04, & n = 2 \\ 0.208, & n = 3 \end{cases} \end{aligned}$$

Also, $y(n) = 0, n < 0$ and for $n \geq 0$,

$$\begin{aligned} y(n) &= v(n) - 0.2v(n-1) \\ &= \begin{cases} 1, & n = 0 \\ 0, & n = 1 \\ 1, & n = 2 \\ 0, & n = 3 \end{cases} \end{aligned}$$

As expected, $y(n) = w(n) = x(n)$ for $n \leq 3$.

(b) Now $h_1(n) = (0.3)^n u(n)$ for all n. Again $v(n) = 0, n < 0$ and

$$v(n) = x(n) + 0.3x(n-1) + 0.09x(n-2) + 0.027x(n-3) + \cdots$$

$$= \begin{cases} 1, & n=0 \\ 0.3, & n=1 \\ 1.09, & n=2 \\ 0.327, & n=3 \end{cases}$$

Again $y(n) = 0, n < 0$ and for $n \geq 0$,

$$\begin{aligned} y(n) &= v(n) - 0.2v(n-1) \\ &= \begin{cases} 1, & n=0 \\ 0.1, & n=1 \\ 1.03, & n=2 \\ 0.109, & n=3 \end{cases} \end{aligned}$$

Evidently, $y(n) \neq x(n)$ for $n > 0$. Nevertheless, since $|y(n) - x(n)| < 0.5$, the decision is correct and $w(n) = x(n), n \leq 3$.

(c) Taking the inverse Z transform of

$$\hat{Y}(z) = \frac{z - 0.2}{z - a} \hat{W}(z),$$

gives the difference equation

$$\forall n, \quad y(n+1) - ay(n) = w(n+1) - 0.2w(n).$$

For $n = 0$ this gives $y(1) - ay(0) = w(1) - 0.2w(0)$, and substituting $y(0) = 1, y(1) = 0.1, w(0) = 1, w(1) = 0$ gives $a = 0.3$, which is the correct estimate of the channel.

4. We have for all n, $h_1(n) = a^n u(n), h_2(n) = \delta(n) + 0.2\delta(n-1)$. From $v = h_1 * x$,

$$v(n) = \begin{cases} 1, & n=0 \\ a, & n=1 \\ 1+a^2, & n=2 \\ a+a^3, & n=3 \end{cases}$$

From $y = h_2 * v$,

$$y(n) = \begin{cases} 1, & n=0 \\ a - 0.2, & n=1 \\ 1 + a^2 - 0.2a, & n=2 \\ a + a^3 - 0.2 - 0.2a^2, & n=3 \end{cases}$$

For $a = 0.6$, this gives $y(0) = 1, y(1) = 0.4, y(2) = 1.24, y(3) = 0.544$. So the decision will be $w(0) = 1, w(1) = 0, w(2) = 1$, but $w(3) = 1 \neq x(3)$.

5. (a) $\hat{H}_1(s) = 1 - \frac{3}{s+1}$, so

$$\forall t, \quad h_1(t) = \delta(t) - 3e^{-3t}u(t).$$

(b) $\hat{H}_2(s) = 1 + 3\frac{3}{s-2}$, so

$$\forall t, \quad h_2(t) = \delta(t) - 3e^{2t}u(-t).$$

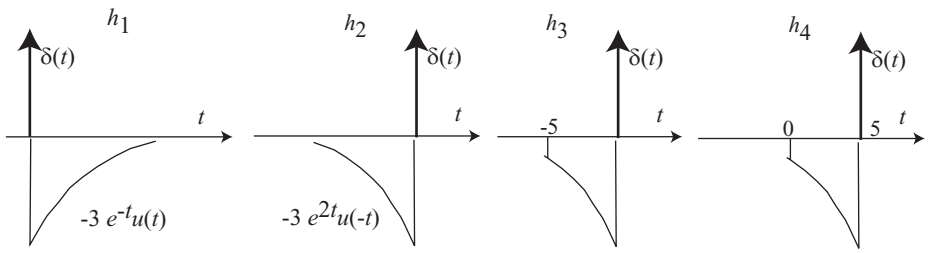


Figure 14.2: Plots for exercise 5

(c) From the definition of the Laplace transform

$$\begin{aligned}\hat{H}_3(s) &= \int_{-5}^{\infty}[\delta(t) - 3e^{2t}u(-t)]e^{-st}dt \\ &= 1 - 3\int_{-5}^{0} e^(2-s)tdt \\ &= 1 + \frac{3}{s-2}[1 - e^{-10}e^{5s}].\end{aligned}$$

The integral above converges absolutely for every s, so *RoC*(h_3) = *Complex*.

(d) Since $h_4(t) = h_3(t-5)$, by the delay property,

$$\hat{H}_4(s) = e^{-5s}\hat{H}_3(s),$$

and *RoC*(h_4) = *RoC*(h_3) = *Complex*. Figure 14.2 sketches the various plots.

6. (a) Let

$$\hat{H}_1(s) = \frac{Ks}{s + K/M},$$

then

$$\hat{H}_2(s)\hat{H}_1(s) = \frac{K/M}{s + K/M},$$

the same as the closed loop transfer function.

(b) The cascade composition has transfer function

$$\hat{H}_2(s)\hat{H}_1(s) = \frac{Ks/M}{(s + K/M)(s - \epsilon)},$$

which is unstable since the pole at $p = \epsilon$ is not cancelled.

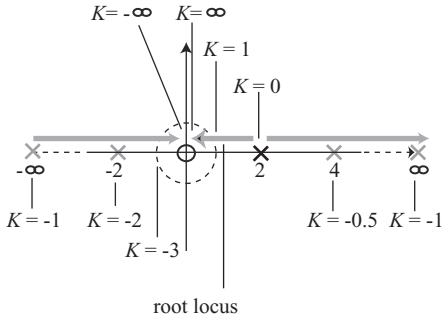
7. (a) There is one zero at 0 and one pole at 2. Since the pole has magnitude greater than 1, the open-loop system is not stable.

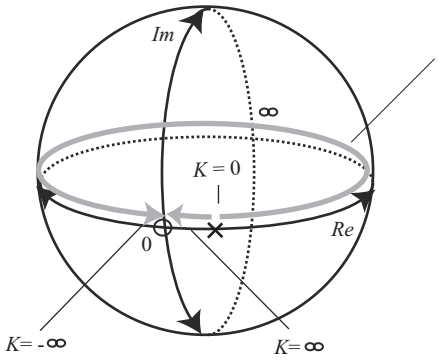
(b) The impulse response is $h(n) = 2^n u(n)$ for all n, and it is unbounded.

(c) The closed loop system transfer function is

$$\begin{aligned}\hat{H}(z) &= \frac{K\hat{H}_2(z)}{1+K\hat{H}_2(z)} \\ &= \frac{K\frac{z}{z-2}}{1+K\frac{z}{z-2}} \\ &= \frac{Kz}{(K+1)z-2} \\ &= \frac{K/(K+1)}{1-(2/(K+1))z^{-1}}\ ,\end{aligned}$$

(d) The single pole of the closed loop system is at $2/(K+1)$. The root locus is as shown below:





At $K = 0$, the closed loop pole is the same as the open loop pole, at 2. As K increases from 0 to $+\infty$ the closed loop pole moves left from 2 to 1 (for $K = 1$), and eventually at $K = +\infty$ the pole reaches 0, the location of the open loop zero. On the other hand, as K decreases from 0 to $-\infty$, the closed loop pole moves right from 2 to 4 (for $K = -0.5$) to ∞ (for $K = -1$); it them moves right, starting at $-\infty$ and eventually at $K = -\infty$ the pole again reaches 0. The discontinuous movement of the pole from $+\infty$ (at $K = -1 + \epsilon$) to $-\infty$ (at $K = -1 - \epsilon$) can be visualized as a continuous movement by imagining the complex plane as a sphere with the origin at the front and infinity at the back, as shown in the figure. The lower part of figure above depicts this view.

(e) For

$$K > 1 \text{ or } K < -3,$$

the pole has magnitude less than 1, and so the closed loop system is stable.

(f) The step input x has Z transform $\hat{X}(z) = z/(z-1)$. The Z transform $\hat{Y}$ of the output y is

$$\hat{Y}(z) = \hat{G}(z)\hat{X}(z) = \frac{Kz}{(K+1)z-2}\frac{z}{z-1} \tag{14.1}$$

$$= \frac{(2K/(1-K^2))z}{z-2/(K+1)} + \frac{(K/(K-1))z}{z-1}\,. \tag{14.2}$$

Taking inverse Z transform gives

$$\forall\, n, \quad y(n) = \frac{2K}{1-K^2}(\frac{2}{K+1})^n u(n) + \frac{K}{K-1}u(n).$$

The first term is the transient response, which vanishes as $n \to \infty$ and the second term is the steady-state response. So

$$\lim_{n\to\infty} y(n) = \frac{K}{K-1}. \tag{14.3}$$

For $K = 10$, the steady-state response is $10/9 \approx 1.11$, so the steady-state tracking error is about 0.11.

(g) With this new plant, the P controller is stable with

$$K > 1+\epsilon \text{ or } K < -2-\epsilon. \tag{14.4}$$

Thus, if choose value of K such that

$$K > 1 + 0.5 \text{ or } K < -2 - 0.5,$$

then the P controller is robust for plants with $|\epsilon| < 0.5$.

8. (a) Write

$$\hat{H}_1\hat{H}_2 = \frac{\hat{A}}{\hat{B}}\,.$$

Since $\hat{H}_1\hat{H}_2$ is strictly proper, the order of $\hat{B}$ exceeds that of $\hat{A}$. The closed-loop transfer function is given by

$$\hat{H} = \frac{\hat{H}_1\hat{H}_2}{1+\hat{H}_1\hat{H}_2} = \frac{\hat{A}}{\hat{B}+\hat{A}}\,.$$

Since the order of $\hat{B}$ exceeds that of $\hat{A}$, the order of the denominator must exceed the order of the numerator in this closed-loop transfer function, so the transfer function is strictly proper.

(b) To see this, write $\hat{H}_1\hat{H}_2$ as a rational polynomial in z,

$$\hat{H}_1(z)\hat{H}_2(z) = \frac{\hat{A}(z)}{\hat{B}(z)} .$$

Since this is strictly proper, the order of the denominator polynomial $\hat{B}(z)$ is strictly larger than the order of the numerator. Let

$$\hat{G}(z) = z\hat{H}_1(z)\hat{H}_2(z) = z\frac{\hat{A}(z)}{\hat{B}(z)} .$$

$\hat{G}(z)$ is proper because the order of $\hat{B}(z)$ is strictly greater than the order of $\hat{A}(z)$, and hence the order of $\hat{B}(z)$ is greater than or equal to the order of $z\hat{A}(z)$.

(c) The significance of this is that the feedback loop contains within it a unit delay, which has transfer function z^{-1}, in cascade with a causal system. The unit delay means that the output of $\hat{H}_1\hat{H}_2$ does not depend on the current input. It only depends on previous inputs, which are reflected in the state. Thus, $\hat{H}_1\hat{H}_2$ is a state-determined system, and the feedback loop is well-formed. Intuitively, this unit delay makes it easy to calculate the output of the feedback loop at each input sample, because the output does not depend on the current input.

(d) To see this, write $\hat{H}_1\hat{H}_2$ as a rational polynomial in s,

$$\hat{H}_1(s)\hat{H}_2(z) = \frac{\hat{A}(s)}{\hat{B}(s)} .$$

Since this is strictly proper, the order of the denominator polynomial $\hat{B}(s)$ is strictly larger than the order of the numerator. Let

$$\hat{G}(s) = s\hat{H}_1(s)\hat{H}_2(s) = s\frac{\hat{A}(s)}{\hat{B}(s)} .$$

$\hat{G}(s)$ is proper because the order of $\hat{B}(s)$ is strictly greater than the order of $\hat{A}(s)$, and hence the order of $\hat{B}(s)$ is greater than or equal to the order of $z\hat{A}(s)$.

(e) An integrator has transfer function s^{-1}, and moreover, has state-determined output. The output of an integrator does not depend on the current input if the input is bounded and piecewise continuous, but rather only depends on previous inputs. Note that with the conditions on the input, the input could contain Dirac delta functions, which would result in the output of the integrator depending on the current input.

9. If p is real and strictly negative $(s-p)$ is a polynomial with real and strictly positive coefficients. If $p = a+ib$ and $p* = a-ib$ and a is strictly negative, $(s-p)(s-p*) = (s-a)^2+b^2$ is a polynomial with real and strictly positive coefficients. The product of such polynomials will also have real and strictly positive coefficients.

10. (a) The closed loop transfer function is

$$\hat{H}(s) = \frac{\hat{H}_1(s)\hat{H}_2(s)}{1+\hat{H}_1(s)\hat{H}_2(s)} = \frac{K_1 s + K_2}{s^3 + K_1 s + K_s}.$$

The denominator polynomial has coefficient of 0 for s^2, hence it must have a root (which is a pole of $\hat{H}$) in the right-half plane. So the system is unstable.

(b) The closed loop transfer function now is

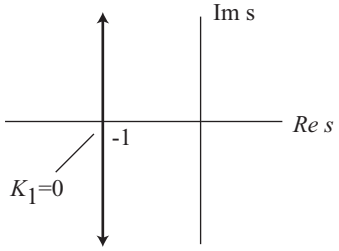
$$\hat{H}(s) = \frac{\hat{H}_1(s)\hat{H}_2(s)}{1 + \hat{H}_1(s)\hat{H}_2(s)} = \frac{K_1 + K_2 s}{s^2 + K_2 s + K_1},$$

whose denominator $s^2 + K_2 s + K_1$ can be set equal to $(s - p_1)(s - p_2)$ for any choice of real p_1, p_2 or complex conjugate p_1, p_2.

11. (a) With $K_2 = 0$, the closed loop transfer function is

$$\hat{H}(s) = \frac{K_1}{s^2 + 2s + (K_1 + 1)},$$

so the closed loop poles are at $s = -1 \pm i\sqrt{K_1}$ (assuming $K_1 \geq 0$). So the closed loop system is stable for all $K_1 \geq 0$. The root locus is shown below:



The response to a unit step input has transfer function $\hat{H}(s) \times 1/s$. Its steady state limit is $K_1/(K_1 + 1)$. So the steady state error is $-1/(K_1 + 1)$.

(b) The closed loop transfer function is

$$\hat{H}(s) = \frac{K_1 s + K_2}{s^3 + 2s^2 + (K_1 + 1)s + K_2},$$

which for $K_1 = K_2 = 1$ has poles at $-1, -0.5 \pm 0.87i$ and so the sytem is stable. There is no steady state error for step inputs.

Appendix A

Sets and Functions

1. Sketches:

2. (a) 6 elements, 1, 2, 3, 4, 5, 6

   (b) 13 elements, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

   (c) 6 elements, (0,2), (0,3), (1,2), (1,3), (2,2), (2,3)

3. (a) False. $1 \in$ *Naturals*.

   (b) True.

   (c) False.

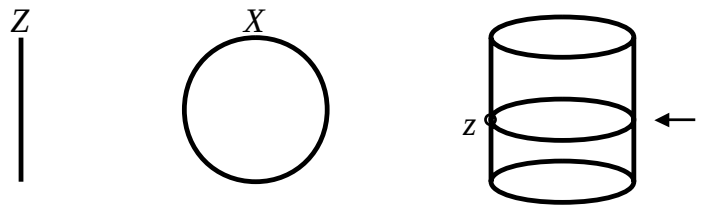
   (d) True.

   (e) True.

4. The product of an m-dimensional set with an n-dimensional set is a new set with $m + n$ dimensions. One way to think of $Z \times X$ is to imagine that each point in the line becomes a circle. That is, each and every element in the 1-dimensional line is combined with every element in the 2-dimensional circle set. The new 3-dimensional set, when graphed, is a cylinder. In symbols,

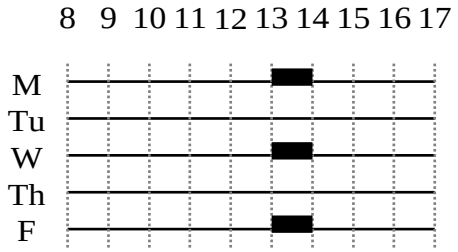
$$Z \times X = \bigcup_{z \in Z} \{z\} \times X$$

   Graphically, the product can be drawn as:



The arrow points to $\{z\} \times X$.

5. Sketch:



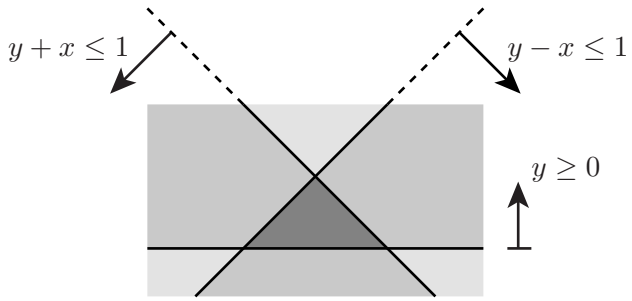
The figure assumes the class is scheduled to meet MWF 11-12.

6. Sketch:

In the second set, both elements of the tuple must be identical. In the first set, they can be different.

7. One of (infinitely) many solutions:

$$\mathit{Pred}(x, y) = (x \geq 0) \wedge (y + x \leq 1) \wedge (y - x \leq 1).$$



8. (a) mn
   (b) $\prod_{i=1}^{I} m_i$.

9. (a) 26^{10}.
   (b) 10×25^9. For each of 10 positions of 'a', the other nine letters can be arbitrarily assigned.

10. (a) $10 \cdot 9/2 = 45$.
    (b) $\frac{n}{[(n-m)!m!]}$.

11. (a) *BigCities* $= \{$*city* $\in$ *USCities* $\mid$ population of city $> 106\}$.
    (b) *MaleStudents* $= \{$*student* $\in$ *ClassStudents* $\mid$ gender of student = male $\}$.
    (c) *OldBooks* $= \{$*book* $\in$ *LibraryBooks* $\mid$ age of book = old$\}$.

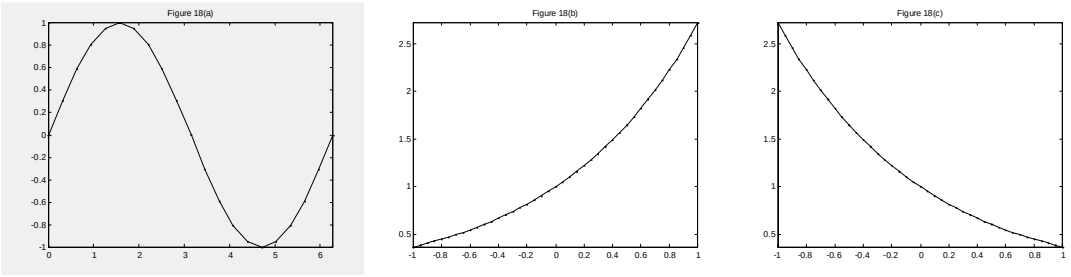
12. (a) Well-formed, false
    (b) Not well-formed
    (c) Well-formed, true
    (d) Not well-formed
    (e) Well-formed, $\{1, 2, 3, 4\}$
    (f) Well-formed, true
    (g) Well-formed, (3, 4)
    (h) Well-formed, true

13. (a) *Char*^{10}
    (b) *Reals*^{30} or $(\textit{Reals}^5)^6$ or $(\textit{Reals}^6)^5$.
    (c) $\{c \in$ *Complex* $\mid |c| \leq 1\}$.
    (d) $\{(v_1, v_2) \in \textit{Reals}^2 \mid v_1^2 + v_2^2 = 1\}$.

14. $\emptyset$, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}

15. 2^n. Any subset $Y \subset X = \{x_1, \ldots, x_n\}$ can be represented as an n-digit binary number $y = (y_1, \cdots, y_n) \in \{0, 1\}^n$ such that $y_i = 1$ or 0 accordingly as x_i is or is not in Y. The set of all possible subsets of X ($P(X)$, the power set of X) can in this way be put in correspondence with the set of all n-digit binary numbers. It follows that the number of elements in $P(X)$ is 2^n.

16. (a) `t=0:(pi/10):(2*pi); plot(t,sin(t));`
    (b) `x=-1:(1/20):1; plot(x,exp(x));`
    (c) `x=-1:(1/20):1; plot(x,exp(-x));`



17. (a) Different vehicles have different license numbers so *License* is one-to-one. However, not every finite string in *Char** is a license number, so it is not onto.
    (b) f is not one-to-one since, for example, $f(x) = f(x + 2\pi)$ for every x. It is onto as can be seen from the graph of the sin function.
    (c) f is not one-to-one as above. It is not onto: for example, there is no x such that $2\sin(x) = 3$.
    (d) *conj* is one-to-one, since *conj*$(x_1 + jy_1) =$ *conj*$(x_2 + jy_2)$ if and only if $x_1 = x_2, y_1 = y_2$. It is onto since for any $(x + jy)$ in the range, *conj*$(x - jy) = x + jy$.
    (e) f is one-to-one and onto.
    (f) From linear algebra M is one-to-one and onto if and only if its determinant is not zero, as in this case.
    (g) *Zero* is not onto (there is no x for which *Zero* $= (1, 1, 1, 1)$), and it is not one-to-one (since *Zero*$(1, 1, 1, 1) =$ *Zero*$(0, 0, 0, 0)$).

18. (a) True. Both expressions describe a set containing all the elements in A or B that are not also not in C.
    (b) False. Take $A = \{1\}$ and $B = \{2\}$. Then $\{1, 2\} \in P(A \cup B)$, but $\{1, 2\} \notin P(A)$ and $\{1, 2\} \notin P(B)$.
    (c) False. We provide a counterexample. Take for example f: *Naturals* $\rightarrow$ *Naturals* such that for all $n \in$ *Naturals*, $f(n) = n + 1$. Take g: *Naturals* $\rightarrow$ *Naturals* such that for all $n \in$ *Naturals*, $g(n) = 2n$. Then, for example, $(f \circ g)(1) = 3$ and $(g \circ f)(1) = 4$, which are not equal.
    (d) True. Consider any x and y in the domain of g such that $x \neq y$. Then $g(x) \neq g(y)$ because g is one-to-one. Then $f(g(x)) \neq f(g(y))$ because f is one-to-one. Hence, $f \circ g$ is one-to-one.

19. (a) false.
    (b) not well formed.
    (c) true.
    (d) free y.

Appendix B

Complex Numbers

1. (a) Note that

$$\frac{3+i4}{5-i6} = \frac{(3+i4)(5+i6)}{25+36} = \frac{-9+i38}{61}.$$

$$\frac{3+i6}{4-i5} = \frac{(3+i6)(4+i5)}{16+25} = \frac{-18+i39}{41}.$$

Hence

$$\frac{3+i4}{5-i6} \times \frac{3+i6}{4-i5} = \frac{-9+i38}{61} \times \frac{-18+i39}{41} = -0.5278 - i0.4138,$$

by Matlab. (Note that the whole simplication could have been done by Matlab, but it's useful to see how to do it by hand.)

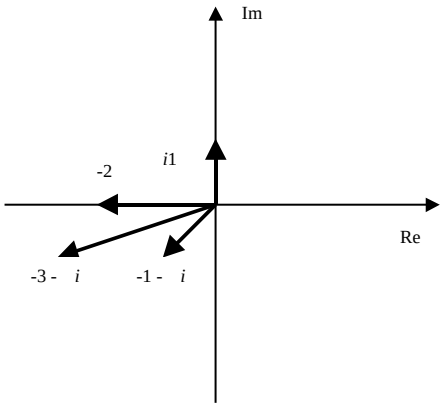
(b)

$$e^{2+\pi i} = e^2 e^{\pi i} = -e^2.$$

2.

$$\begin{aligned}
(2-2i) &= 2\sqrt{2}e^{-i\pi/4}, \\
(2+2i) &= 2\sqrt{2}e^{i\pi/4}, \\
1/(2-2i) &= 1/(2\sqrt{2}e^{-i\pi/4}) \approx 0.35e^{i\pi/4}, \\
1/(2+2i) &= 0.35e^{-i\pi/4}, \\
2i &= 2e^{i\pi/2}, \\
-2i &= 2e^{-i\pi/2}.
\end{aligned}$$

3. The plot is:



4. Note that

$$1 + i = \sqrt{2}e^{i\pi/4}.$$

So we can write

$$\begin{aligned} Re\{(1+i)e^{i\theta}\} &= \sqrt{2}Re\{e^{i\pi/4}e^{i\theta}\} \\ &= \sqrt{2}Re\{e^{i(\theta+\pi/4)}\} \\ &= \sqrt{2}\cos(\theta + \pi/4). \end{aligned}$$

So our task is to find θ so that

$$\sqrt{2}\cos(\theta + \pi/4) = -1.$$

So

$$\theta = \cos^{-1}(-1/\sqrt{2}) - \pi/4.$$

There are two values for $\cos^{-1}(-1/\sqrt{2})$ in the range 0 to 2π, namely $3\pi/4$ and $5\pi/4$, which yields the two possible solutions

$$\theta = \pi/2 \quad \text{and} \quad \pi.$$

We can check both of these by substituting back into the original equation.

5. To find 6 distinct solutions to $z^6 = 1$, we first write $z = re^{i\theta}$ in polar form, where $r \geq 0$ is a nonnegative real number. We want the 6 solutions to

$$r^6 e^{i6\theta} = 1.$$

Noting that the magnitude must be equal on both sides,

$$r^6 = 1.$$

Since r is a nonnegative real number, it must be true that $r = 1$. So all 6 solutions have magnitude 1. Thus, we need to find 6 distinct values of θ so that

$$e^{i6\theta} = 1 = e^{i0}.$$

Hence, we need

$$6\theta \bmod 2\pi = 0$$

The following values all work,

$$\theta = 0, \theta = \pi/3, \theta = 2\pi/3, \theta = \pi, \theta = 4\pi/3, \text{ and } \theta = 5\pi/3.$$

So the 6 roots of unity are $e^{i\theta}$ for each of these values of θ. The next value, $\theta = 2\pi$, results in the same solution as $\theta = 0$.

6. To find 6 distinct solutions to $z^6 = -1$, we first write $z = re^{i\theta}$ in polar form, where $r \geq 0$ is a nonnegative real number. We want the 6 solutions to

$$r^6 e^{i6\theta} = -1.$$

Noting that the magnitude must be equal on both sides,

$$r^6 = 1.$$

Since r is a nonnegative real number, it must be true that $r = 1$. So all 6 solutions have magnitude 1. Thus, we need to find 6 distinct values of θ so that

$$e^{i6\theta} = -1 = e^{i\pi}.$$

Hence, we need

$$6\theta \bmod 2\pi = \pi$$

The following values all work,

$$\theta = \pi/6, \theta = \pi/2, \theta = 5\pi/6, \theta = 7\pi/6, \theta = 9\pi/6, \text{ and } \theta = 11\pi/6.$$

So the 6 roots of unity are $e^{i\theta}$ for each of these values of θ. The next value, $\theta = 13\pi/6$, results in the same solution as $\theta = \pi/6$.

7. We calculate the positive exponents first

$$\begin{aligned} i^0 &= 1 \\ i^1 &= \sqrt{-1} \\ i^2 &= -1 \\ i^3 &= -\sqrt{-1} \\ i^4 &= 1 \\ i^n &= i^{n-4} \quad \forall\, n \geq 5. \end{aligned}$$

We calculate the negative exponents next

$$\begin{aligned} i^{-1} &= 1/i = i/i^2 = -\sqrt{-1} \\ i^{-2} &= 1/i^2 = -1 \\ i^{-3} &= \sqrt{-1} \\ i^{-4} &= 1 \\ i^{-n} &= i^{-n+4} \quad \forall\, n \geq 5. \end{aligned}$$

8. Find 5 roots of $z^5 + 2$ by solving

$$z^5 + 2 = 0$$

or

$$z^5 = -2.$$

Let $z = re^{i\theta}$ be the polar form representation, and solve

$$r^5 e^{i5\theta} = -2 = 2e^{i\pi}.$$

Since the magnitudes on both sides must be equal, and r is a nonnegative real number,

$$r = 2^{(1/5)}.$$

Moreover, the arguments must be equal, modulo 2π,

$$5\theta \bmod 2\pi = \pi$$

so

$$\theta = \pi/5,\ 3\pi/5,\ \pi,\ 7\pi/5,\ \text{or}\ 9\pi/5.$$

So the solutions are $re^{i\theta}$ with the specified value of r and values of θ.

9. Let $x = \sqrt{1+i}$ mean any x such that $x^2 = 1 + i = \sqrt{2}e^{i\pi/4}$. So,

$$x = 2^{(1/4)}e^{i\pi/8} \quad \text{or} \quad 2^{(1/4)}e^{i(\pi+\pi/8)}.$$

Just as with the square root of real numbers, there are two answers. Alternatively, it is defensible to define $\sqrt{1+i}$ to be only the first of these, since the second is the negative of the first.

More generally, define $x = \sqrt{z}$ to be any x satisfying $x^2 = z$. Write $z = re^{i\theta}$ in polar form, so

$$x = \sqrt{r}e^{i\theta/2} \quad \text{or}\sqrt{r}e^{i(\pi=\theta/2)}.$$

Again, we could take the first of these to be the definition, since the second is the negative of the first.

10. By the given definition, $\log 1$ is all w satisfying

$$e^w = 1$$

so

$$\log 1 = 0,\ i2\pi,\ i4\pi,\ \cdots.$$

Similarly, $\log -1$ is all w satisfying

$$e^w = -1$$

so

$$\log -1 = i\pi,\ i3\pi,\ i5\pi,\ \cdots.$$

Similar methods yield

$$\log i = i\pi/2,\ i5\pi/2,\ i9\pi/2,\ \cdots.$$

and

$$\log -i = i7\pi/2,\ i5\pi/2,\ i11\pi/2,\ \cdots.$$

Finally, $\log(1+i)$ is all w satisfying

$$e^w = 1 + i = \sqrt{2}e^{i\pi/4}.$$

Write this as

$$e^{Re\{w\}}e^{iIm\{w\}} = \sqrt{2}e^{i\pi/4},$$

so

$$Re\{w\} = \log(\sqrt{2}),$$

where this is the ordinary natural logarithm, and

$$Im\{w\} = \pi/4 + 2\pi n$$

for all $n \in$ *Integers*. More generally, if $z = re^{j\theta}$, then

$$Re\{\log(z)\} = \log(r),$$

and

$$Im\{\log(z)\} = \theta + 2\pi n$$

for all $n \in$ *Integers*.

11. (a) Note in the following that Matlab accepts both i and j for imaginary numbers.

```
>> z1 = 2+3j;
>> z2 = 4-2i;
>> z = z1 + z2

z =

   6.0000 + 1.0000i

>> real(z)

ans =

     6

>> imag(z)

ans =

     1
```

(b) Note that

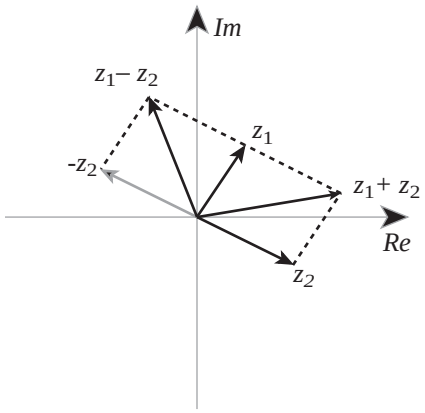


Figure B.1: Sum and difference of two complex numbers.

```
>> abs(z)

ans =

    6.0828

 angle(z)

ans =

    0.1651
```

Thus,

$$z = 6.0828e^{0.1651i}.$$

(c) The plot is shown in figure B.1. The sum can be systematically constructed by placing the two rays for the terms being summed head to tail, in either order. The difference can be constructed by reflecting the subtracted term and then summing.

(d)
```
>> z3 = -2-3i;
>> z4 = 3-3i;
>> z3*z4

ans =

 -15.0000 - 3.0000i

>> z3/z4

ans =
```

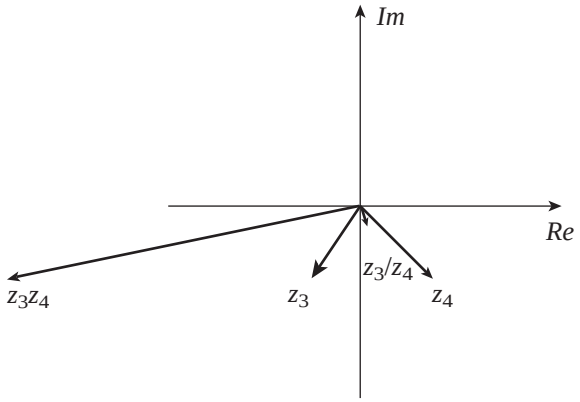


Figure B.2: Two complex numbers, their product, and their quotient.

```
0.1667 - 0.8333i
```

These are shown in figure B.2.

(e) Let $z5 = 2e^{i\pi/6}$. Then $z6 = z5^* = 2e^{-i\pi/6}$. Multiplication of complex numbers in polar form is easy: multiply the magnitudes and add the exponents. In this case $z5z6 = 4$. Note that if z is a complex number then $|z|^2 = zz^*$. In Matlab,

```
>> z5 = 2*exp(pi/6 * 1i);
>> z6 = conj(z5);
>> z5*z6

ans =

     4
```

(f) Since we are going to multiply complex numbers lets express $z0$ in polar form. We find that $z_0 = \sqrt{2}e^{i\pi/4}$. Thus $z_n = \sqrt{2}e^{i(\pi/4+n\pi/4)}$. Note that for any real x and any integer n we have that $e^{ix} = e^{i(x+2\pi n)}$. Thus from here we see that there will be 8 distinct z_n's. The sketch is shown in figure B.3.

(g) We want to find all distinct z such that $z^7 = 1$. Now, recall that any polynomial of order n has at most n distinct solutions. We'll show that in our case we find exactly 7 distinct solutions.

Let $z = re^{i\theta}$. Then $z^7 = (r^7)e^{i7\theta}$. We can also express 1 as $1e^{2\pi n}$ for any n. Thus by comparing magnitudes of z^7 and 1 we see that $r = 1$. Now, we can choose $\theta = 2\pi n/7$ for $n = 0, 1, 2, 3, 4, 5, 6$.

Check that $z_n = e^{2\pi n/7}$ for $n = 0, .., 6$ are all distinct. Also check that $z_7 = z_0$, $z_8 = z_1$ etc ...

12. (a) The plot is shown in figure B.4, with the purely imaginary values marked with 'x.' No, it is not evident what values of t yield purely imaginary $f(t)$.

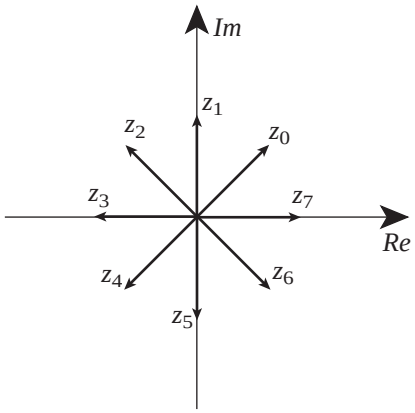


Figure B.3: Complex number rotation.

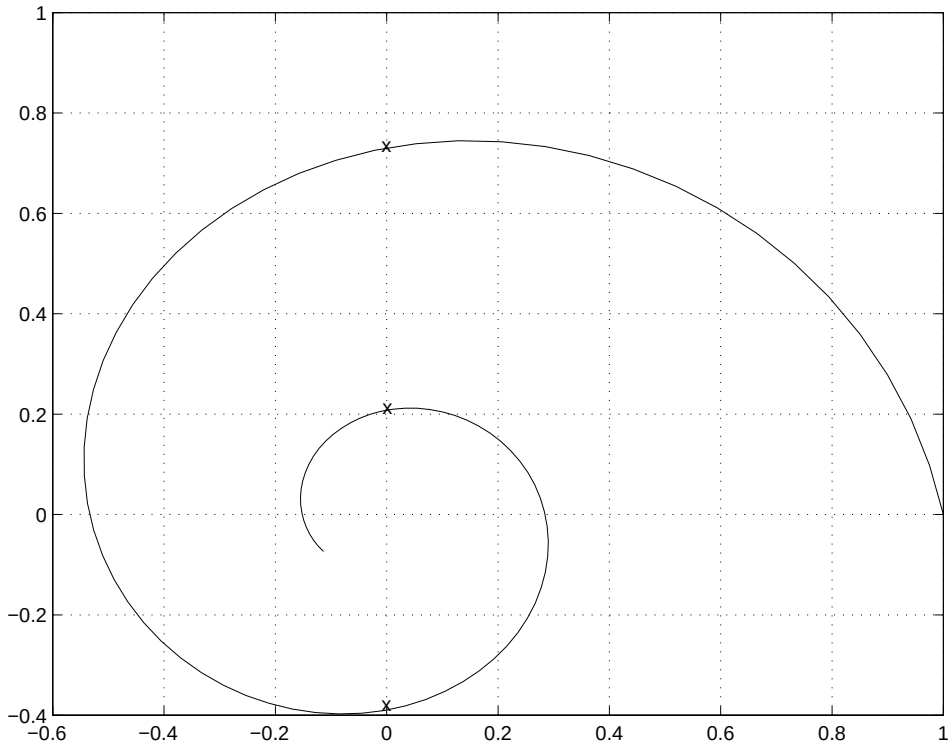


Figure B.4: Result of `plot(exp((-2+10i)*[0:0.01:1]))`.

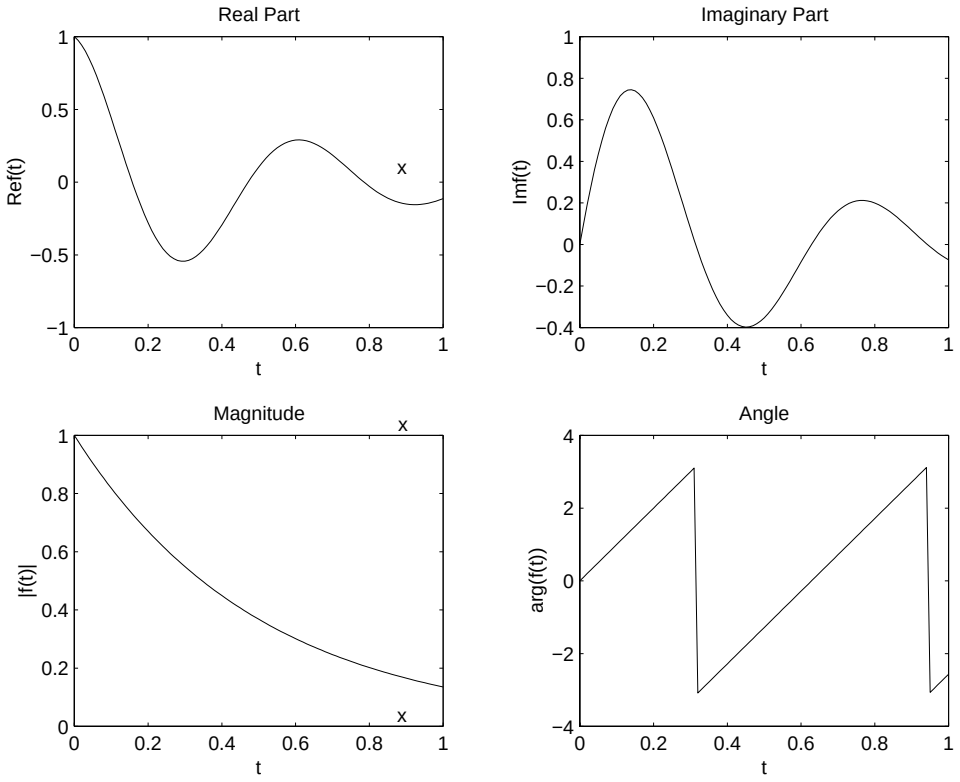


Figure B.5: Real, imaginary, magnitude, and angle of the function plotted in figure B.4.

(b) Express $f(t)$ in polar form, $f(t) = e^{-2t}e^{i10t}$. From here we readily identify that any t with $10t = \pi n$ (for some integer n) will yield $f(t)$ purely real, so values $t = \pi n/10$ for all integers n result in purely real $f(t)$. Purely imaginary $f(t)$ result when $10t = \pi n + \pi/2$ for some integer n. Thus, values $t = \pi n/10 + \pi/20$ result in purely imaginary $f(t)$.

(c) The four plots are shown in figure B.5. They are generated by the Matlab commands:

```
>> t = [0:0.01:1];
>> x = exp((-2+10i)*t);
>> subplot(2,2,1); plot(t, real(x));
>> subplot(2,2,2); plot(t, imag(x));
>> subplot(2,2,3); plot(t, abs(x));
>> subplot(2,2,4); plot(t, angle(x));
```

(d) The expressions for the magnitude and angle of f follow readily from its polar form representation: $|f(t)| = e^{-2t}$ and $\angle f(t) = 10t$.

Note that if we want to keep $\angle f$ in $[-\pi, \pi]$ or in $[0, 2\pi]$ then we should write the expression accordingly (i.e. modulo 2π).

Now, to obtain the expression for the real and imaginary part simply recall that if z is a

complex number then $real(z) = (z + z^*)/2$ and $imag(z) = (z - z^*)/2i$. Thus,

$$\begin{aligned} f(t) &= e^{-2t}e^{i10t} \\ Re\{f(t)\} &= e^{-2t}\cos(10t) \\ Im\{f(t)\} &= e^{-2t}\sin(10t) \\ |f(t)| &= e^{-2t} \\ \angle f(t) &= 10t. \end{aligned}$$

13. (a) Using the given hint we write $e^{i2\theta} = (e^{i\theta})^2$, so

$$\cos(2\theta) + i\ \sin(2\theta) = \cos(\theta)^2 - \sin(\theta)^2 + i\,2\cos(\theta)\sin(\theta)$$

The result follows by comparing the real and imaginary parts of this equality:

$$\cos(2\theta) = \cos(\theta)^2 - \sin(\theta)^2$$

and

$$\sin(2\theta) = 2\cos(\theta)\sin(\theta).$$

(b) Using the same procedure as in part (a) we obtain

$$\cos(3\theta) + i\ \sin(3\theta) = (\cos(\theta)^2 - \sin(\theta)^2 + i\,2\cos(\theta)\sin(\theta)) \cdot (\cos(\theta) + i\ \sin(\theta))$$

and simplifying we obtain:

$$\cos(3\theta) = \cos(\theta)^3 - 3\cos(\theta)\sin(\theta)^2$$

and

$$\sin(3\theta) = 3\cos(\theta)^2\sin(\theta) - \sin(\theta)^3.$$

(c) If we solve for A, ϕ in the equation

$$Ae^{i(\omega t+\phi)} = \sum A_k e^{i(\omega t+\phi_k)} \tag{B.1}$$

we obtain the desired answer by taking the real part of this equation. Multiplying both sides of (B.1) with $e^{-i\omega t}$ gives the simpler equation

$$Ae^{i\phi} = \sum A_k e^{i\phi_k} = \sum A_k \cos\phi_k + i\sum A_k \sin\phi_k.$$

The left-hand side is in polar coordinates and the right-hand side is in Cartesian coordinates. So

$$\begin{aligned} A &= [(\sum A_k \cos\phi_k)^2 + (\sum A_k \sin\phi_k)^2]^{1/2} \\ \phi &= \tan^{-1}\frac{\sum A_k \sin\phi_k}{\sum A_k \cos\phi_k}. \end{aligned}$$

Appendix C

Laboratory Exercises Solutions

C.1 Arrays and sound solution

C.1.1 In-lab section

1. (a) `[1 2 3 4 5]` is 1×5, a row vector. `[1:5]` is 1×5, a row vector. `1:5` is 1×5, a row vector. `[1:1:5]` is 1×5, a row vector. `[1:-1:-5]` is 1×7, a row vector. `[1 2; 3 4]` is 2×2, a square matrix. `[1; 2; 3; 4]` is 4×1, a column vector.

   (b) A constant 2×3 matrix:

   ```
   M = [11 12 13 ; 21 22 23]
   ```

   Matrices can also be constructed piecemeal. Here is row-by-row and column-by-column construction:

   ```
   row1 = [ 11 12 13 ]; row2 = [ 21 22 23 ];
   mat = [ row1 ; row2 ]
   col1 = [ 11 ; 21 ]; col2 = [ 12 ; 22 ]; col3 = [ 13 ; 23 ];
   mat = [ col1 col2 col3 ]
   ```

   We can also do this with expressions that return matrices:

   ```
   [ rand(1,3) ; ones(1,3) ]
   [ zeros(2,1) randn(2,1) ones(2,1) ]
   ```

   Here is a matrix constructed by putting together a column vector and a square matrix:

   ```
   [ [ rand(1) ; pi ] eye(2) ]
   ```

   (c)
   ```
   >> size(1:0.3:10)

   ans =

        1    31
   ```

```
>> size(1:1:-1)

ans =

     1     0
```

The `size` function gives both the number of rows and the number of columns. Notice that in the second case, there are no columns, so this can be interpreted as an empty row vector. You can confirm this interpretation with the `length` function:

```
>> length(1:1:-1)

ans =

     0
```

Now, the array constructor pattern

```
array = start : step : stop
```

returns an array of numbers between `start` and `stop` that begin with `start` and are incremented by `step`. If `stop` is less than `start`, then a zero-length array is returned. Otherwise, the result is an array of length

```
floor((stop-start)/step) + 1
```

2. (a) Here is a for loop that computes the sum:

```
>> x = 0
for i = 0:25
x = x + i;
end


x =

     0

>> x

x =

   325
```

Notice that if we leave off the semicolon above we will see all the partial sums.

(b)
```
>> sum(0:25)

ans =

   325
```

(c)
```
>> sin(0:pi/10:pi)

ans =

  Columns 1 through 7

    0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511

  Columns 8 through 11

    0.8090    0.5878    0.3090    0.0000
```

(d)
```
>> n = sin(0:pi/10:pi);
>> n.*n

ans =

  Columns 1 through 7

    0    0.0955    0.3455    0.6545    0.9045    1.0000    0.9045

  Columns 8 through 11

    0.6545    0.3455    0.0955    0.0000
```

Note that `n^2` will not work (it triggers an error message). A vector cannot be squared in this way.

3. (a)
```
>> x = zeros(36);
>> x(18) = 1;
>> subplot(2,1,1), plot(x)
>> subplot(2,1,2), stem(x)
```
produces the plots shown in figure C.1.

(b) The frequency is 5 Hz, or $2\pi \times 5$ radians per second. The period is 1/5 second, and there are 10 cycles in the 2-second interval $[-1, 1]$. The value at zero is 0.5.

(c) The following sequence of commands yields the plot in figure C.2.

```
>> frequency = 8000;     % sampling frequency in Hertz
>> period = 1/frequency; % sampling period in seconds
>> t = -1:period:1;      % sample times as a row vector
>> sampledSignal = sin(t,2*pi*5*t + pi/6);
>> plot(sampledSignal)
```

Notice that in the `plot` command we specify both the sample times and the values of the signal. This results in a plot with the horizontal axis correctly labeled in units of

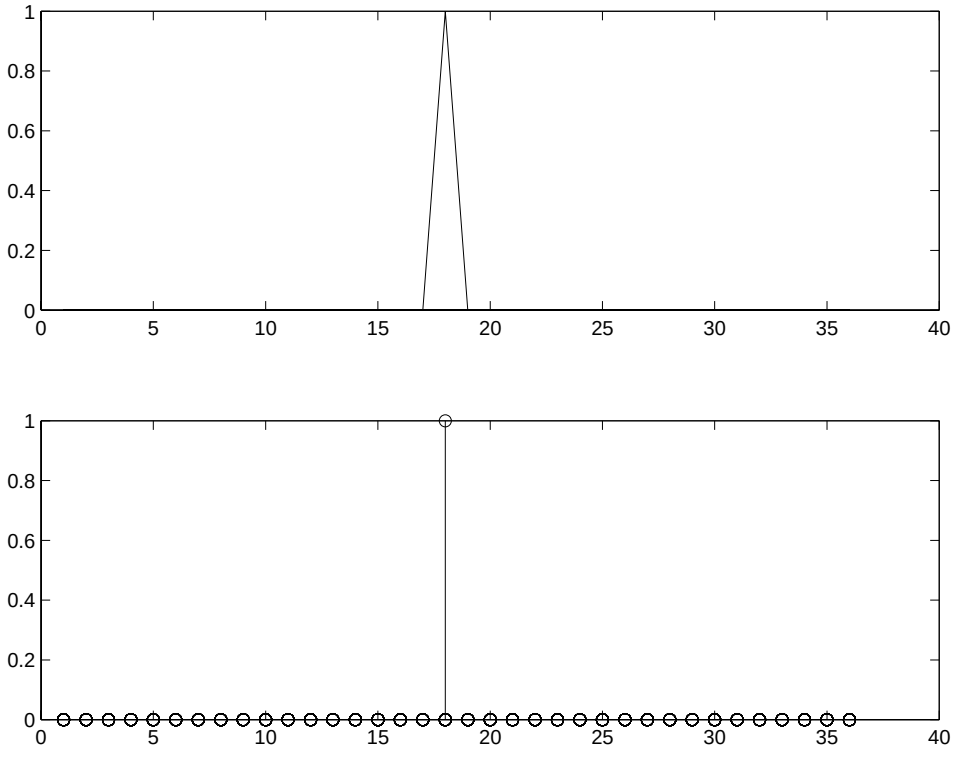


Figure C.1: Two plots of an impulse, generated by `plot` and `stem`.

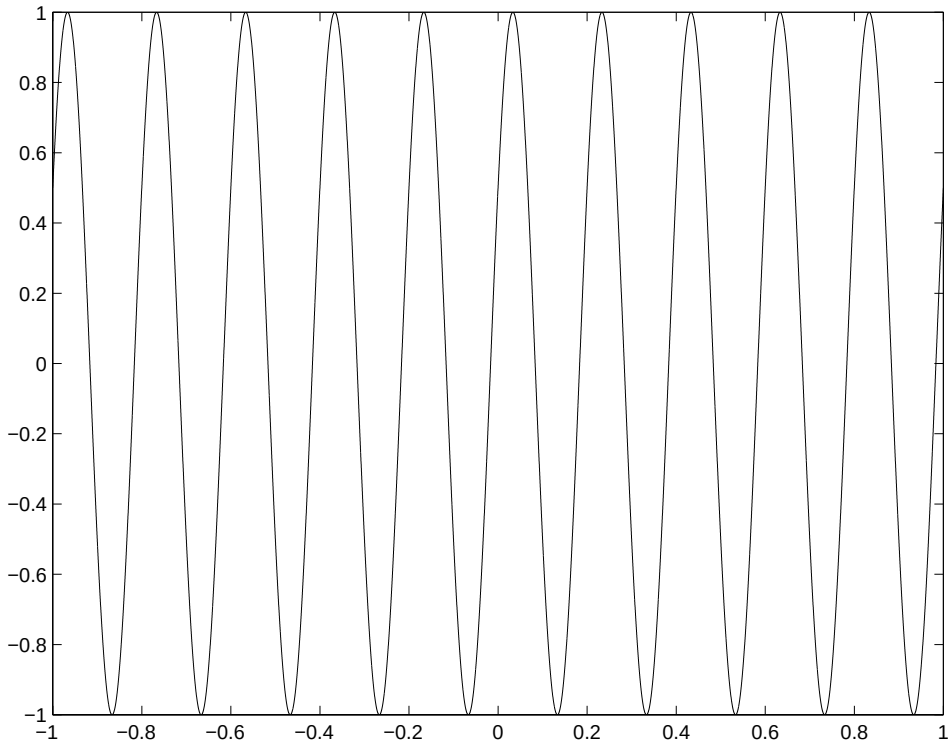


Figure C.2: A sinewave at 5 Hz sampled at 8,000 samples/second.

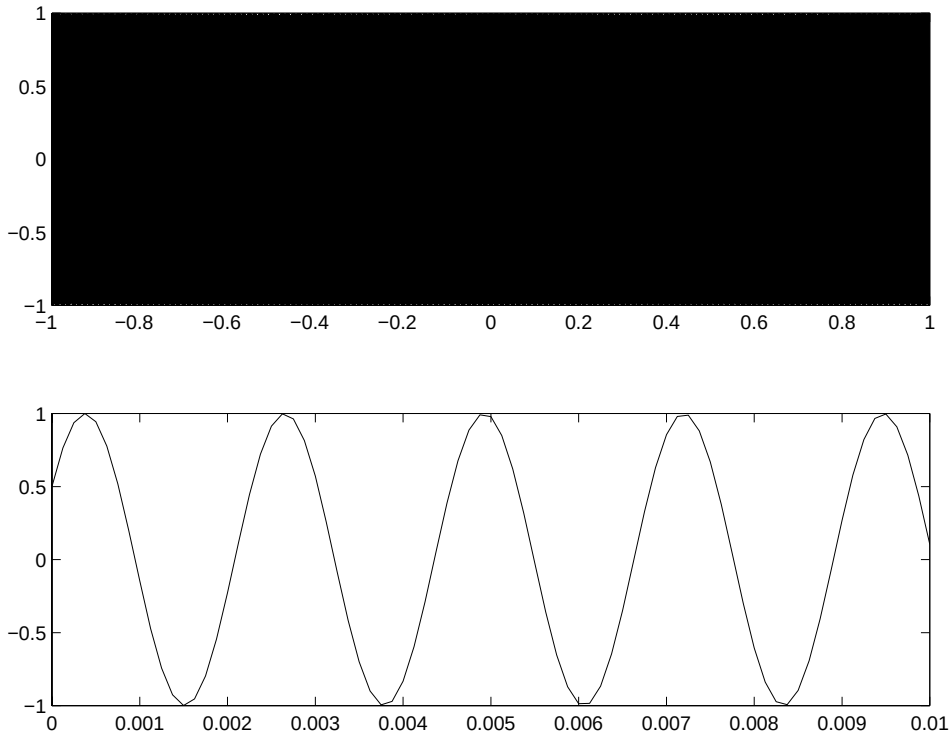


Figure C.3: A sinewave at 440 Hz sampled at 8,000 samples/second.

seconds. There are a total of 16,000 samples. Also notice that all commands end in semicolons to prevent Matlab from displaying the 16,000 samples.

(d) Assuming we have executed the commands above, the following commands result in the plots shown in figure C.3.

```
>> sampledSignal = sin(2*pi*440*t + pi/6);
>> subplot(2,1,1), plot(t, sampledSignal);
>> t = 0:period:0.01;
>> sampledSignal = sin(2*pi*440*t + pi/6);
>> subplot(2,1,2), plot(t, sampledSignal);
```

The upper plot is very hard to read because there are so many cycles that they all run together. The lower plot shows far fewer cycles.

(e) For the 10 msec sound segment, you hear just a click. This interval is very short. For the 2 second interval, you hear a tone with frequency 440 Hz, middle A in the western musical scale.

(f) The following command:

```
sound(0.5*sampledSignal,frequency)
```

results in a sound that is quieter, while

```
sound(2*sampledSignal,frequency)
```

is louder and distorted. It is no longer a pure tone. The sound produced by

```
sound(sampledSignal,frequency*2)
```

has a higher pitch, while

```
sound(sampledSignal,frequency/2)
```

has a lower pitch.

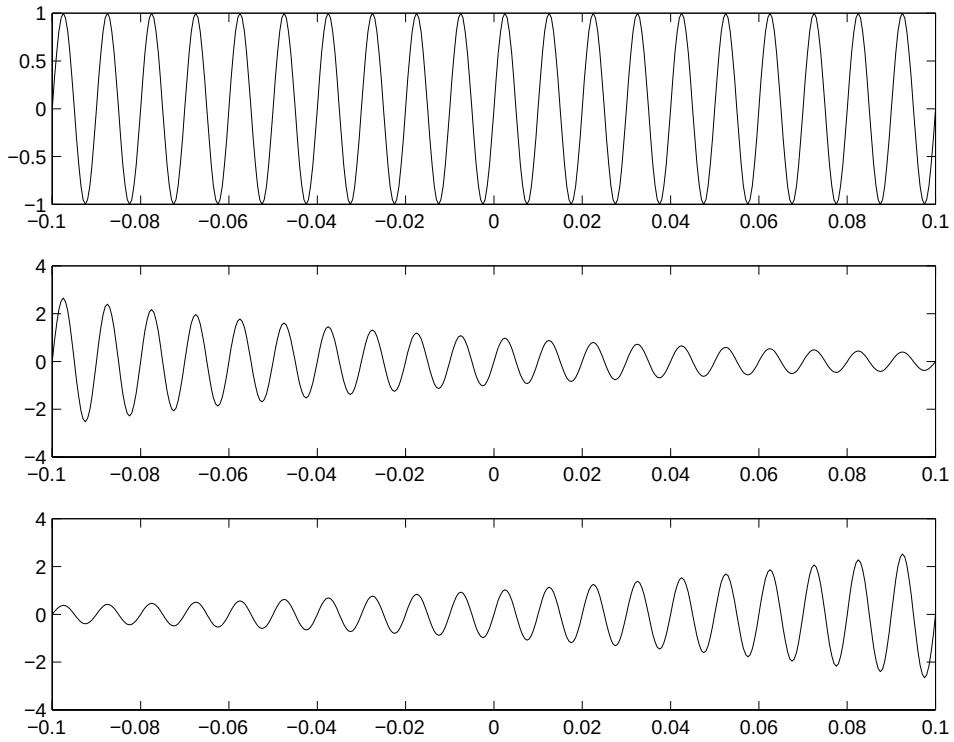


Figure C.4: Three sinusoidal signals with exponential envelopes.

C.1.2 Independent section

1. The sinusiodal term in each of these signals completes one cycle in 0.01 seconds. To get a reasonably smooth plot, we will need several samples in each cycle, say 20. Thus, we set

```
>> t = -0.1:0.01/20:0.1;
>> y1 = sin(2*pi*100*t);
>> subplot(3,1,1), plot(t,y1)
>> y2 = exp(-10*t).*sin(2*pi*100*t);
>> subplot(3,1,2), plot(t,y2)
>> y3 = exp(10*t).*sin(2*pi*100*t);
>> subplot(3,1,3), plot(t,y3)
```

   The resulting plot is shown in figure C.4.

2. The specified sound is constructed as follows:

```
t = 0:1/8000:1;
n = exp(-5*t).*sin(2*pi*440*t);
sound(n, 8000);
```

It is a middle A (A-440), but with a sudden attack and a gradual, exponential decay.

3. The following program constructs the requested sound:

```
t = 0:1/8000:0.5;
n = []
for frequency = [494 440 392 440 494 494 494]
   n1 = exp(-5*t).*sin(2*pi*frequency*t);
   n = [n n1];
end
sound(n, 8000);
```

It is the first few notes of "Mary had a little lamb."

4. (a)

$$\mathit{sound}\colon \mathit{Vectors} \to \mathit{Sounds}$$

where *Vectors* is the set of row or column vectors of elements of *Double*. The set *Vectors* can be defined as follows:

$$\mathit{Vectors} = \{v \in \mathit{Doubles}^N \mid N \in \mathit{Naturals}\}.$$

(b) *soundsc* has the same domain and range.

(c) The plot function generates a picture, so ignoring color,

$$\mathit{plot}\colon [\mathit{Naturals} \to \mathit{Doubles}] \to [\mathit{HorizontalSpace} \times \mathit{VerticalSpace} \to \mathit{Intensity}].$$

C.2 Images solution

C.2.1 In-lab section

1. White is represented as [1.0, 1.0, 1.0] and black as [0.0, 0.0, 0.0].

2. The following script does the job:

```
>> k = 0:199;
>> x = (sin(k*2*pi/200 + pi/2) +1)';
>> b = 128 * repmat(x, 1, 200);
>> image(b), axis image
```

   Note that the transpose operator (the apostrophe on the end of the second line) converts a row vector into a column vector. This could also have been done with for loops, but the result is much less elegant and very much slower to execute:

```
>> for row = 1:200
      for col = 1:200
         b(row,col) = 128*(1 + cos(2*pi*row/200));
      end
   end
>> image(b), axis image
```

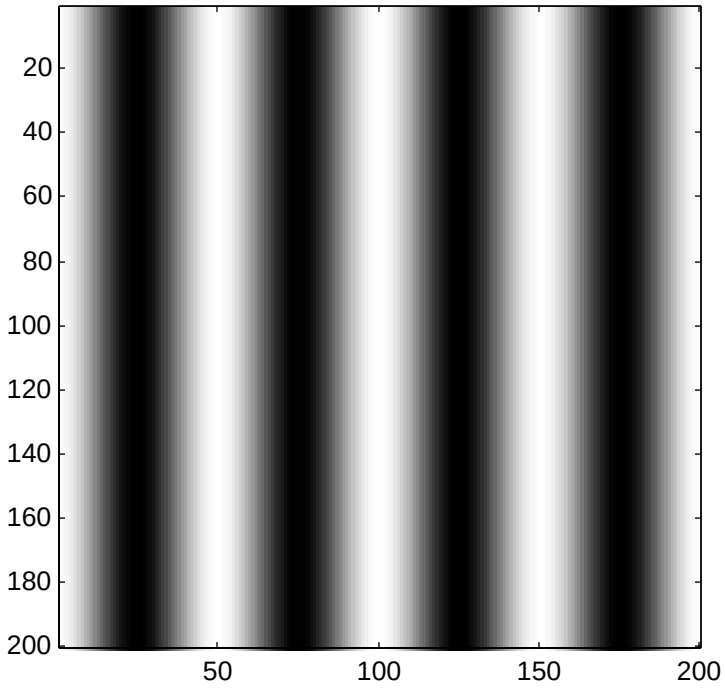
3. The following script does the job:

```
>> k = 0:199;
>> x = (sin(k*2*pi/50 + pi/2) +1);
>> c = 128 * repmat(x, 200, 1);
>> image(c), axis image
```

   This differs from the previous in that there is no transpose on the second line and the frequency higher (divide by 50 rather than 200). Also, the `repmat` repeats a row rather than a column. Again, an inelegant solution using for loops is:

```
>> for row = 1:200
      for col = 1:200
         c(row,col) = 128*(1 + cos(2*pi*col/50));
      end
   end
>> image(c), axis image
```

   The image is shown below:



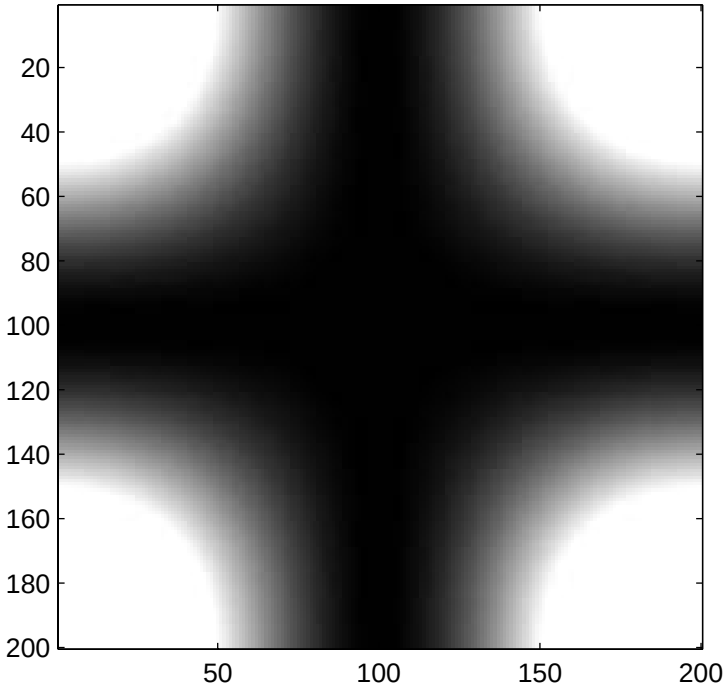
4. Using the above definitions of b and c, the following is sufficient:

```
>> d = b * c;
>> image(d), axis image
```

Or equivalently,

```
>> for row = 1:200
      for col = 1:200
         sinImage(row,col) = 128 *(1 + cos(2*pi*col/200)) ...
 * (1 + cos(2*pi*row/200));
      end
   end
>> image(sinImage), axis image
```

The "..." indicates to Matlab that the command is continued on the next line. The result is shown below:



5. The `whos` command yields:

```
>> whos helen
  Name        Size         Bytes  Class

  helen     300x200x3     180000  uint8 array
```

This specifies that the `helen` array is $300 \times 200 \times 3$, from which I would infer that no colormap is used. This is a 300×200 pixel color image with the RGB values given for each pixel. This is what is meant by `'truecolor'`. Colormap images are said to be in pseudocolor. The size in bytes is 180,000, compared to a file size of 18,026. This means that the compression ratio is about 10 to 1.

6. To look at the lower right pixel:

```
>> squeeze(helen(300,200,:))

ans =

    21
    26
    56
```

In the image, we see that the upper right pixel is very light and the lower right is very dark. From this, we infer that white must be [255, 255, 255] and black must be [0, 0, 0] in truecolor images.

C.2.2 Independent section

1. The `image` function that we have been using can be defined as:

$$\textit{MatlabImage}: \textit{ColormapImages} \times \textit{Colormaps} \rightarrow \textit{Images}$$

where

$$\textit{ColormapImages} = [\{1, \cdots, 200\} \times \{1, \cdots, 200\} \rightarrow \{1, \cdots, 256\}]$$

and

$$\textit{Colormaps} = [\{1, \cdots, 256\} \rightarrow [0, 1]^3]$$

and

$$\textit{Images} = [\textit{DiscreteHorizontalSpace} \times \textit{DiscreteVerticalSpace} \rightarrow \textit{Intensity}^3].$$

2. The following program creates a movie:

```
k = 0:199;
numFrames = 15;
m = moviein(numFrames);
for frame = 1:numFrames
   x = sin(k*2*pi/200 + pi/2 + (frame - 1)*2*pi/numFrames) + 1;
   b = repmat(x', 1, 200) * 128;
   image(b), axis image
   m(:,frame) = getframe;
end
movie(m)
```

Again, an inelegant solution using for loops looks like this:

```
numFrames = 15;
m = moviein(numFrames);
for frame = 1:numFrames;
   for row = 1:200
      for col = 1:200
         sinImage(row,col) = 128*(1 + cos(2*pi*row/200 ...
         + (frame-1)*2*pi/numFrames));
      end
   end
   image(sinImage)
   axis image
   m(:,frame) = getframe;
end
movie(m)
```

In this case, the for loops are dramatically slower. You should avoid them when possible. The command

```
movie(m, 100, 30)
```

repeats the movie 100 times at 30 frames per second (if the computer is fast enough).

3. To get the colormap for the red separation, we do

```
>> mask = [ones(256,1), zeros(256,2)];
>> redcolormap = map.*mask;
>> colormap(redcolormap)
```

which results in the image shown at the left in figure C.5. The green color separation is generated by

```
green = helen(:,:,2);
image(green), axis image
mask = [zeros(256,1), ones(256,1), zeros(256,1)];
greencolormap = map.*mask;
colormap(greencolormap)
```

which results in the image shown in the center in figure C.5. The blue color separation is generated by

```
blue = helen(:,:,3);
image(blue), axis image
mask = [zeros(256,2), ones(256,1)];
bluecolormap = map.*mask;
colormap(bluecolormap)
```

which results in the image shown at the right in figure C.5.

4. The blurred image of figure C.6 is constructed as follows:

```
blurred = zeros(300,200);
for row = 3:298
   for col = 3:198
      avg = 0;
      for i = -2:2
         for j = -2:2
            avg = avg + bwImage(row-i, col-j);
         end
      end
      blurred(row,col) = avg/25;
   end
end
image(blurred), axis image
```

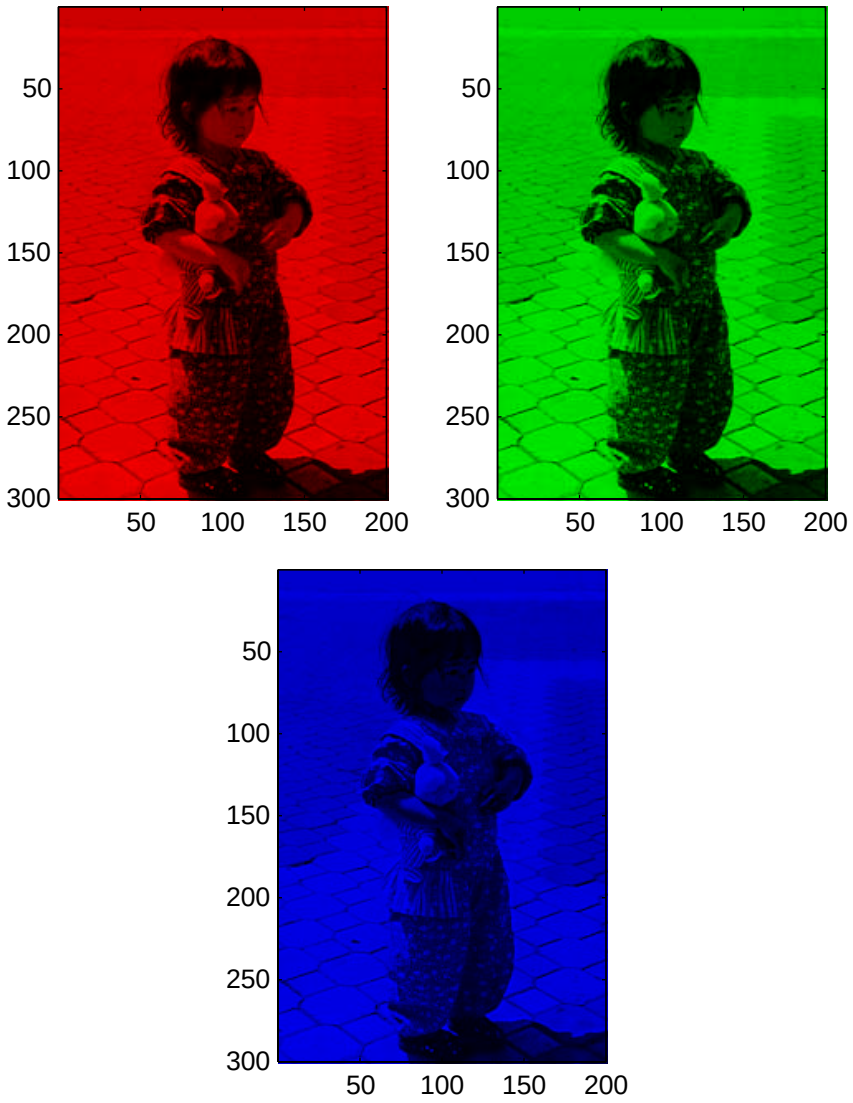


Figure C.5: Color separations in red, green, and blue.

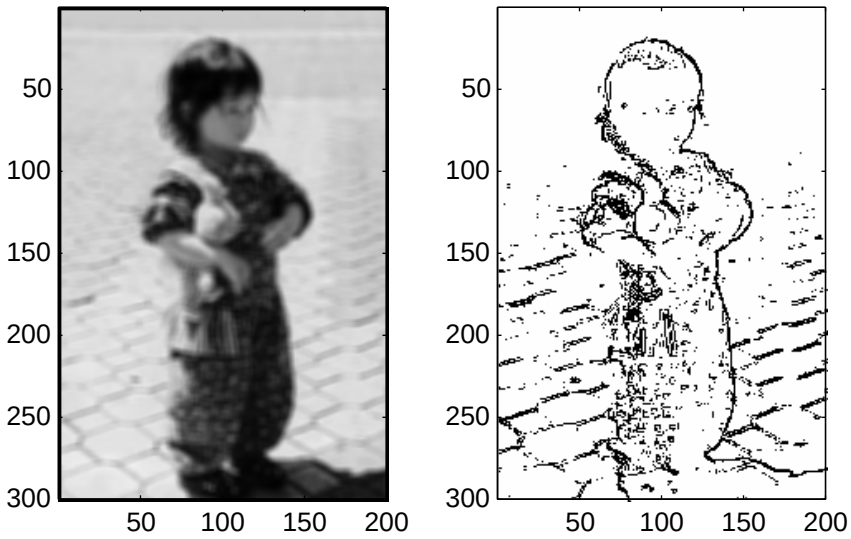


Figure C.6: Blurred image constructed with a 5×5 moving average (left), and simple edge detection (right).

5. The following program results in the plot shown on the right in figure C.6.

```
edges = ones(size(bwImage))*255;
threshold = 40;
for row = 2:300
   for col = 2:200
      vertDiff = bwImage(row, col) - bwImage(row - 1, col);
      horDiff = bwImage(row, col) - bwImage(row, col - 1);
      if ((abs(vertDiff) > threshold) | (abs(horDiff) > threshold))
         edges(row,col) = 0;
      end
   end
end
image(edges), axis image
```

The threshold of 40 yields a reasonable compromise between detail and clutter.

C.3 State machines

C.3.1 Background

No exercises here.

C.3.2 In-lab section

1.
```
>> d = {'a' 'b' 'a' 'a' 'b'};
>> n = 0;
>> for e = d
      if strcmp(e, 'a') n = n+1;
      end
   end
>> n

n =

     3
```

Note how the for loop works. The variable `e` gets each element of `d` each time through the loop. The function definition looks like this:

```
function result = count(arg)
% COUNT - Count the number of occurrences of 'a' in the argument.
result = 0;
for e = arg
   if strcmp(e, 'a') result = result + 1;
   end
end
```

Then

```
count(['a', 'b', 'c', 'a', 'aa'])

ans =

     4
```

and

```
>> count({'a', 'b', 'c', 'a', 'aa'})

ans =
```

```
     2
```

Notice the difference between these. In the first instance, an array is the argument with one character forming each element of the array. In the second instance, a cell array is the argument with a string forming each element. Thus, the result of the count is different in the two cases.

2. The following program does the job:

```
% COUNTAS - Count the number of a's in the input.
while 1
    response = input('Enter a string:','s');
    if (strcmp(response,'quit') | ...
            strcmp(response,'exit'))
        break;
    end
    disp(count(response));
end
```

The 's' argument to the `input` function indicates that the input should be interpreted as a string, rather than being evaluated as a Matlab expression. The "..." indicates to Matlab that the line is continued.

3. Here is a definition of the update function:

```
function [next, output] = update(state, input)
% UPDATE - simple state machine update.
% The state argument should be 0 or 1.
% The input argument should be '0', '1', or 'absent'.
% The output result is '0', '1', or 'absent'.
% The next result is 0 or 1.
switch input
case '0'
    next = 0;
    if state output = '1';
    else output = '0';
    end
case '1'
    next = 1;
    if state output = '1';
    else output = '0';
    end
otherwise
    next = state;
    output = 'absent';
end
```

Notice that any state that is non-zero is interpreted as 1. Any input that is not 0 or 1 is interpreted as absent. Here is a program that executes this state machine in response to user input:

```
% DELAY - invoke the update function repeatedly.
state = 0;
while 1
    in = input('Enter 0, 1, absent, or quit: ', 's');
    if (strcmp(in, 'quit') | strcmp(in, 'exit'))
        break;
    end
    [state, output] = update(state, in);
    disp(output);
end
```

C.3.3 Independent section

1. The alphabets are

$$Inputs = \{feed, pet, time\ passes, absent\}$$

$$Outputs = \{purrs, bites, throws\ up, rubs, dies, absent\}$$

The state machine is shown in figure C.7. The update of the state machine is given by the following function

```
% PET - A function representing the state update of a virtual pet.
% The first argument must be in {'happy', 'hungry', 'dies'}
% The second argument must be in {'absent', 'pet', 'feed', 'time passes'}
% The two returned values are the next state of the
% pet, and the output of the state machine.

function [newstate, out] = pet(state, in)

% The default behavior is to stutter.
newstate = state;
out = 'absent';

switch(state)
case 'happy'
    switch(in)
    case 'pet'
        out = 'purrs';
    case 'feed'
        out = 'throws up';
    case 'time passes'
        newstate = 'hungry';
        out = 'rubs';
    end
case 'hungry'
    switch(in)
    case 'feed'
        out = 'purrs';
        newstate = 'happy';
    case 'pet'
        out = 'bites';
    case 'time passes'
        out = 'dies';
        newstate = 'dies';
    end
case 'dies'
    out='dies';
end
```

A program to execute the state machine is:

```
% RUNPET - Execute the virtual pet state machine

% initial state.
petstate='happy';

% loop forever.
while 1
    % Get the user input as a string.
    str=input('enter one of absent, pet, feed, time passes: ','s');

    % If the user entered quit or exit, then break the loop.
    if strcmp(str,'quit') break; end
    if strcmp(str,'exit') break; end

    % Update the pet.
    [petstate, output] = pet(petstate, str);

    % print what the pet does.
    disp(output)
    if strcmp(petstate, 'dies') break; end
end
```

Here is a sample run:

```
>> runpet
enter one of absent, pet, feed, time passes: pet
purrs
enter one of absent, pet, feed, time passes: feed
throws up
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: feed
purrs
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: pet
bites
enter one of absent, pet, feed, time passes: time passes
dies
>>
```

2. The output alphabet for the driver machine is

$$Outputs' = \{feed, time\ passes\}$$

which is a subset of *Inputs*, the input alphabet for the pet. Thus, this machine can be composed in cascade with the pet. The state machine is shown in figure C.8. The state mirror those of the pet with the same name.

The update of the state machine is given by the following function

```
% DRIVER - A function representing the state update of
% a state machine providing inputs to keep a virtual pet alive.
% The first argument must be in {'happy', 'hungry'}
% The second argument must be in {'absent', '1'}
% The two returned values are the next state of the
% driver, and the output of the state machine.

function [newstate, out] = driver(state, in)

% Alternate producing 'time passes' and 'feed'.
if ~strcmp(in, 'absent')
    switch(state)
    case 'happy'
        out = 'time passes';
        newstate = 'hungry';
    otherwise
        out = 'feed';
        newstate = 'happy';
    end
else
    % The default behavior is to stutter.
    newstate = state;
    out = 'absent';
end
```

A program to run the driver and pet for 100 iterations is

```
% DRIVEPET - Execute the virtual pet state machine composed
% in cascade with the driver state machine.

% initial state.
driverstate='happy';
petstate='happy';

% loop 100 times, since this is automatically driven.
for i=1:100,
    % Update the state of the driver and get its output.
    % The input to the driver is always '1'.
    [driverstate, petinput] = driver(driverstate, '1');
```

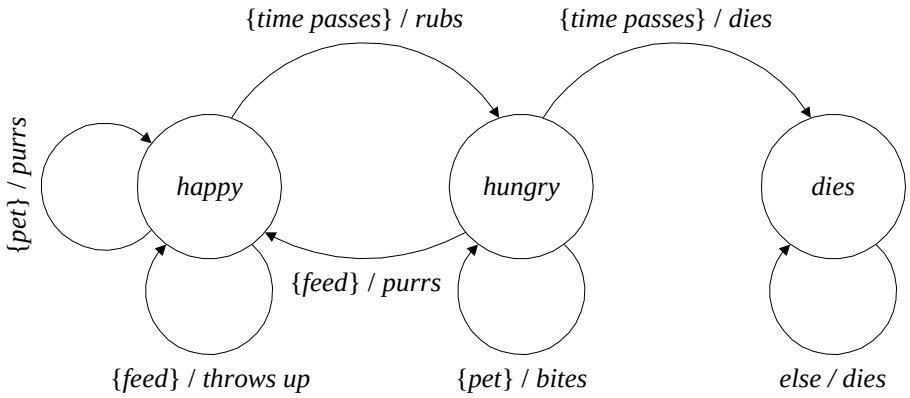


Figure C.7: A state machine for a virtual pet.

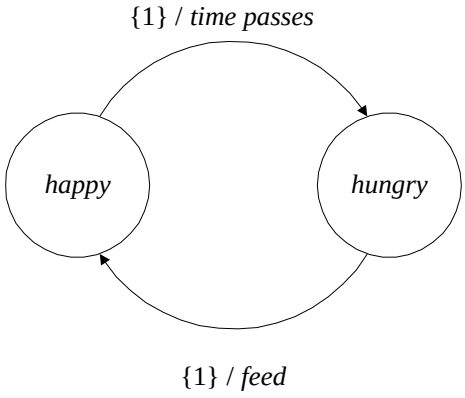


Figure C.8: A state machine that provides inputs to a virtual pet.

```
    % update the state of the pet and get its output.
    [petstate, output] = pet(petstate, petinput);

    % Display the output of the pet.
    disp(output)
end
```

A sample run is

```
>> drivepet
rubs
purrs
rubs
purrs
rubs
purrs
...
```

C.4 Control Systems Solution

C.4.1 In-lab section

1. The following m-file does the job:

```
function selection = select(array)

% SELECT - Given a cell array, randomly select one
% element and return it.

% Generate a random index.
index = floor(1 + rand * length(array));

% There is a small chance that the index will be too
% large (if rand happens to return 1.0), so we have to
% check.
if (index == (length(array) + 1)) index = length(array); end

selection = array(index);
```

   This can be used as follows:

```
>> for i=1:10 x(i) = select(letters); end
>> x

x =

    'a'    'a'    'd'    'c'    'e'    'c'    'c'    'e'    'c'    'b'
```

   The following m-file does the job:

2.
```
function r = chooserow(a)
% COMPOSE - return a randomly chosen row of the
% argument cell array.

% Get the size of the argument.
[n, m] = size(a);

% Generate a random index.
index = floor(1 + rand * n);

% There is a small chance that the index will be too
% large (if rand happens to return 1.0), so we have to
% check.
```

```
if (index == (n + 1)) index = n; end

r = a(index,:);
```

Here we apply it several times:

```
chooserow(t)

ans =

    'upper left'    'upper right'

 chooserow(t)

ans =

    'lower left'    'lower right'

 chooserow(t)

ans =

    'upper left'    'upper right'
```

3. Here is a modified *update* function that returns a cell array:

```
% PETUPDATES - A function representing the state update of a virtual pet.
% The first argument must be in {'absent', 'pet', 'feed', 'time passes'}
% The second argument must be in {'happy', 'hungry', 'dies'}
% The returned value is a two-element cell array, where the first
% element next state of the pet, and the second element
% is the output of the state machine.

function r = pet(state, in)

% The default behavior is to stutter.
r = {state, 'absent'};

switch(state)
case 'happy'
    switch(in)
    case 'pet'
        r = {state, 'purrs'};
    case 'feed'
        r = {state, 'throws up'};
```

```
    case 'time passes'
        r = {'hungry', 'rubs'};
    end
case 'hungry'
    switch(in)
    case 'feed'
        r = {'happy', 'purrs'};
    case 'pet'
        r = {state, 'bites'};
    case 'time passes'
        r = {'dies', 'dies'};
    end
case 'dies'
    r = {state, 'dies'};
end
```

Here is a modified program to run the pet (without the driver):

```
% RUNPET - Execute the virtual pet state machine

% initial state.
petstate='happy';

% loop forever.
while 1
    % Get the user input as a string.
    str=input('enter one of absent, pet, feed, time passes: ','s');

    % If the user entered quit or exit, then break the loop.
    if strcmp(str,'quit') break; end
    if strcmp(str,'exit') break; end

    % Update the pet.
    r = petUpdates(petstate, str);
    petstate = r{1};

    % print what the pet does.
    disp(r{2})
    if strcmp(petstate, 'dies') break; end
end
```

4. Only one line changes:

```
% PETUPDATES - A function representing the state update of a virtual pet.
% The first argument must be in {'absent', 'pet', 'feed', 'time passes'}
```

```
% The second argument must be in {'happy', 'hungry', 'dies'}
% The returned value is a two-element cell array, where the first
% element next state of the pet, and the second element
% is the output of the state machine.

function r = pet(state, in)

% The default behavior is to stutter.
r = {state, 'absent'};

switch(state)
case 'happy'
    switch(in)
    case 'pet'
        r = {state, 'purrs'};
    case 'feed'
        r = {state, 'throws up'};
    case 'time passes'
        r = {'hungry', 'rubs'};
    end
case 'hungry'
    switch(in)
    case 'feed'
        r = {'happy', 'purrs'; 'hungry', 'rubs'};
    case 'pet'
        r = {state, 'bites'};
    case 'time passes'
        r = {'dies', 'dies'};
    end
case 'dies'
    r = {state, 'dies'};
end
```

In the program that runs the cat, again, only one line changes:

```
% RUNPET - Execute the virtual pet state machine

% initial state.
petstate='happy';

% loop forever.
while 1
    % Get the user input as a string.
    str=input('enter one of absent, pet, feed, time passes: ','s');
```

```
    % If the user entered quit or exit, then break the loop.
    if strcmp(str,'quit') break; end
    if strcmp(str,'exit') break; end

    % Update the pet.
    r = chooserow(petUpdates(petstate, str));
    petstate = r{1};

    % print what the pet does.
    disp(r{2})
    if strcmp(petstate, 'dies') break; end
end
```

Here is a sample run:

```
>> runpet
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: feed
purrs
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: feed
rubs
enter one of absent, pet, feed, time passes: feed
rubs
enter one of absent, pet, feed, time passes: feed
rubs
enter one of absent, pet, feed, time passes: feed
rubs
enter one of absent, pet, feed, time passes: feed
purrs
```

5. Here is a modification of the cascade composition that uses the nondeterminstic cat instead of the deterministic one:

```
% DRIVEPET - Execute the virtual pet state machine composed
% in cascade with the driver state machine.

% initial state.
driverstate='happy';
petstate='happy';

% loop 10 times, since this is automatically driven.
for i=1:10,
```

```
    % Update the state of the driver and get its output.
    % The input to the driver is always '1'.
    [driverstate, petinput] = driver(driverstate, '1');

    % update the state of the pet and get its output.
    r = chooserow(petUpdates(petstate, petinput));
    petstate = r{1};

    % Display the output of the pet.
    disp(r{2})
end
```

Here is a typical run:

```
>> drivepet
rubs
purrs
rubs
purrs
rubs
purrs
rubs
rubs
dies
dies
```

Inevitably, we get two 'rubs' in a row and the cat dies.

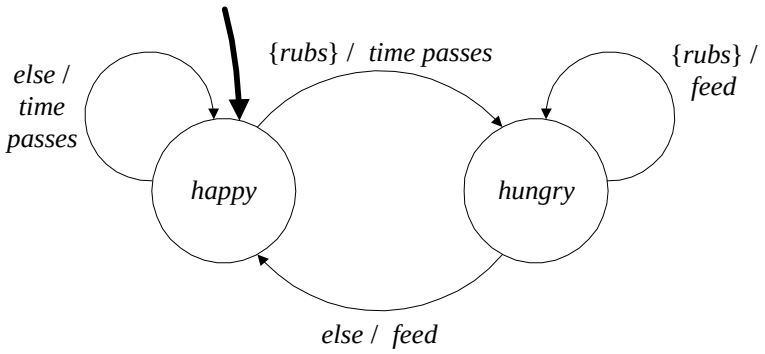


Figure C.9: Controller that keeps the nondeterministic cat alive.

C.4.2 Independent section

The state machine for the controller is shown in figure C.9. Notice that the output that is produced depends only on the state, not on the input. Here is the *output* function of the controller:

```
% CONTROLLEROUTPUT - Given the state of the controller,
% return its output.
function output = controllerOutput(state)

switch(state)
case 'happy'
    output = 'time passes';
otherwise
    output = 'feed';
end
```

The state update of the controller, of course, does depend on the input. The *update* function for the controller is given below:

```
% CONTROLLER - A function representing the state update of
% a state machine providing inputs to keep a virtual pet alive.
% The first argument must be in {'happy', 'hungry'}
% The second argument can be output from the pet.
% The returned value is a 1x2 cell array with the
% next state and the output.

function r = controller(state, in)

if ~strcmp(in, 'absent')
    switch(state)
    case 'happy'
```

```
        if strcmp(in, 'rubs')
            r = {'hungry', 'time passes'};
        else
            r = {'happy', 'time passes'};
        end
    otherwise
        if strcmp(in, 'rubs')
            r = {'hungry', 'feed'};
        else
            r = {'happy', 'feed'};
        end
    end
else
    % The default behavior is to stutter.
    r = {state, 'absent'};
end
```

The program that drives the controller and the nondeterministic cat in a feedback loop is:

```
% DRIVELOOP - Execute the virtual pet state machine composed
% in a feedback loop with the controller state machine.

% Set the initial states.
controllerstate='happy';
petstate='happy';

% loop 10 times, since this is automatically driven.
for i=1:10,
    % Determine the output of the controller.
    petinput = controllerOutput(controllerstate);

    % update the state of the pet and get its output.
    r = chooserow(petUpdates(petstate, petinput));
    petstate = r{1};
    petoutput = r{2};

    % Update the state of the controller.
    % Ignore its output since we've already processed it.
    r = controller(controllerstate, petoutput);
    controllerstate = r{1};

    % Display the output of the pet.
    disp(petoutput)
end
```

Here is a typical output:

```
>> driveloop
rubs
purrs
rubs
purrs
rubs
purrs
rubs
rubs
purrs
rubs
```

C.5 Difference Equations Solutions

C.5.1 In-lab section

1. (a) $M^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $M^1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $M^2 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$, and $M^3 = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}$, so we guess that

$$M^n = \begin{bmatrix} 1 & n \\ 0 & 1 \end{bmatrix}.$$

(b) Putting our guess to the test:

```
M = [1 1; 0 1];
M^25

ans =

     1    25
     0     1
```

we see that Matlab's answer matches our guess.

(c) The guess holds for $n = 0$. Suppose it holds for some fixed n. I.e., for that particular value of n,

$$M^n = \begin{bmatrix} 1 & n \\ 0 & 1 \end{bmatrix}.$$

Multiplying by M we get

$$M^{n+1} = M^n M = \begin{bmatrix} 1 & n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & n+1 \\ 0 & 1 \end{bmatrix}.$$

Thus, the guess holds for $n + 1$. Therefore, it must hold for all integers $n \geq 0$.

2. (a)
```
>> b = [2; 3]

b =

     2
     3

>> A = [1, 1; 0, 1]

A =

     1     1
     0     1
```

```
>> b*A
??? Error using ==> *
Inner matrix dimensions must agree.

>> A*b

ans =

     5
     3
```

The product bA is nonsensical because b has only one column and A has two rows.

(b)
```
>> b'*A

ans =

     2     5

>> A*b'
??? Error using ==> *
Inner matrix dimensions must agree.
```

The product Ab^T is nonsensical because A has two columns but b^T has only one row. The pattern is that the number of columns of the first factor must match the number of rows of the second.

3. The following Matlab function calculates output of the specified system given any input sequence.

```
function y = boing(x, sigma, omega)
% BOING - Return the output of a system with
% an impulse response that is a sinusoid with
% frequency omega (in radians per second) with
% a decaying exponential envelope. Arguments:
%   x     - A vector representing the input sequence.
%   sigma - A scalar determining the rate of decay
%           of the impulse response.
%   omega - A scalar giving the frequency of oscillation
%           in cycles per sample.
A = sigma*[cos(omega), -sin(omega); sin(omega), cos(omega)];
b = [0; 1];
c = sigma*[-cos(omega); sin(omega)];
% Initial state is zero
s = [0; 0];
for i=1:length(x)
```

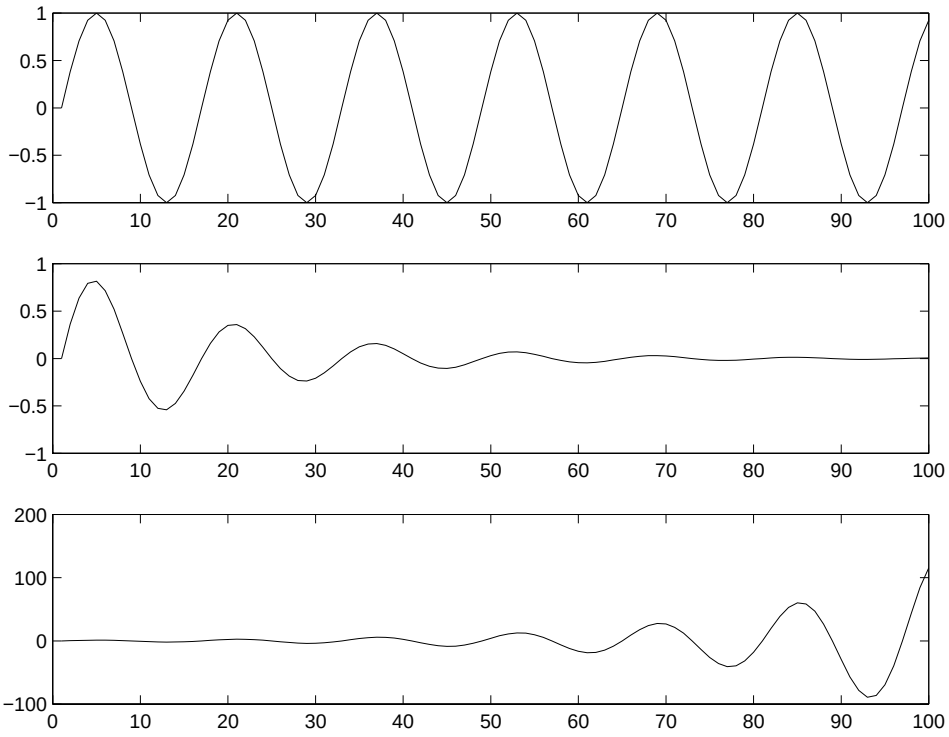


Figure C.10: Solutions to exercise 3 of the in-lab portion.

```
    y(i) = c'*s;
    s = A*s + b*x(i);
end
```

Notice how this code is written. The current state is stored in the variable `s`, which gets update *after* calculating the output inside the loop.

To use this to get the first 100 samples of the zero-state impulse response, we first construct the input sequence, and then calculate and plot it for each of the specified parameters:

```
x = [1, zeros(1,99)];
subplot(3,1,1); plot(boing(x, 1, pi/8));
subplot(3,1,2); plot(boing(x, 0.95, pi/8));
subplot(3,1,3); plot(boing(x, 1.05, pi/8));
```

The results are shown in figure C.10. The impulse response is a sinusoid when $\sigma = 1$, and a sinusoid multiplied by an exponential in the other two cases.

For part (d), we note that only the top result, where $\sigma = 1$, is periodic. The period is 16 samples.

For part (e), the system with $\sigma = 0.95$ is stable; the one with $\sigma = 1.05$ is unstable; and the one with $\sigma = 1.0$ is marginally stable. The amplitude of the unstable system output will grow

very rapidly until it eventually exceeds the numerical range representable by Matlab. It blows up.

C.5.2 Independent section

1. The form of the desired response is the same as that in problem 3 of the in-lab section, so A, b, c, and d are given there. The impulse response is given by

$$h(n) = \begin{cases} d, & \text{if } n = 0 \\ c^T A^{n-1} b, & \text{if } n \geq 1 \end{cases}$$

   which for the specified values of A, b, c, and d can be written

$$h(n) = \sigma^n \sin(\omega n).$$

   To get the desired impulse response, we simply set

$$\omega = 2\pi 440/8000$$

   (which has units of radians per sample), and

$$\sigma = \exp(-5/8000).$$

   We can use the same function defined in the in-lab section. To get 8000 samples of the impulse response and play it as a sound we simply do

```
x = [1, zeros(1,7999)];
sound(boing(x, exp(-5/8000), 2*pi*440/8000));
```

2. To construct the input, we simply do:

```
x = repmat([1, zeros(1,1599)], 1, 10);
sound(boing(x, exp(-5/8000), 2*pi*440/8000));
```

3. In lab C.1 we calculated one second of sound using the Matlab script

```
t = 0:1/8000:1;
n = exp(-5*t).*sin(2*pi*440*t);
sound(n, 8000);
```

   We will count only the operations in the middle line, and we will assume that 8000, rather than 8001 samples are computed. Each sample requires one evaluation of `exp` and one evaluation of `sin`, for a total of 40 multiplications and 30 additions. Each argument requires one multiplication, and the results are multiplied, so there are 43 multiplications per sample. For 8000 samples, that implies $8000 \times 43 = 344000$ multiplications and $8000 \times 30 = 240000$ additions.

   For the state machine model, examining the `boing` function, we see that each output sample involves the calculations given by

```
y(i) = c'*s;
s = A*s + b*x(i);
```

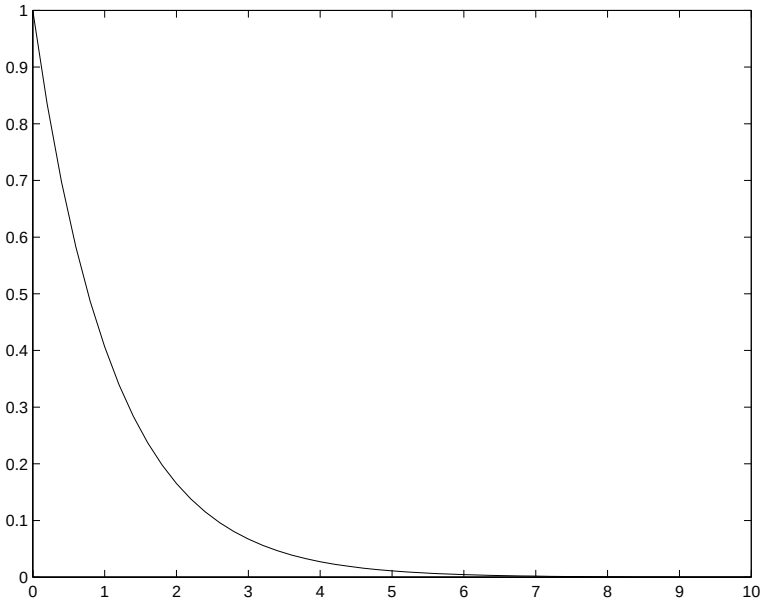
The first line is two multiplications and one addition. The second line is 6 multiplications and two additions. The total is therefore 8 multiplications and 3 additions per sample. For 8000 samples, that implies $8000 \times 8 = 64000$ multiplications and $8000 \times 3 = 24000$ additions. The state machine realization is considerably less expensive by this measure.

C.6 Differential Equations Solutions

C.6.1 In-lab section

1. $$a = -0.9.$$

2. The result of the run is shown below:



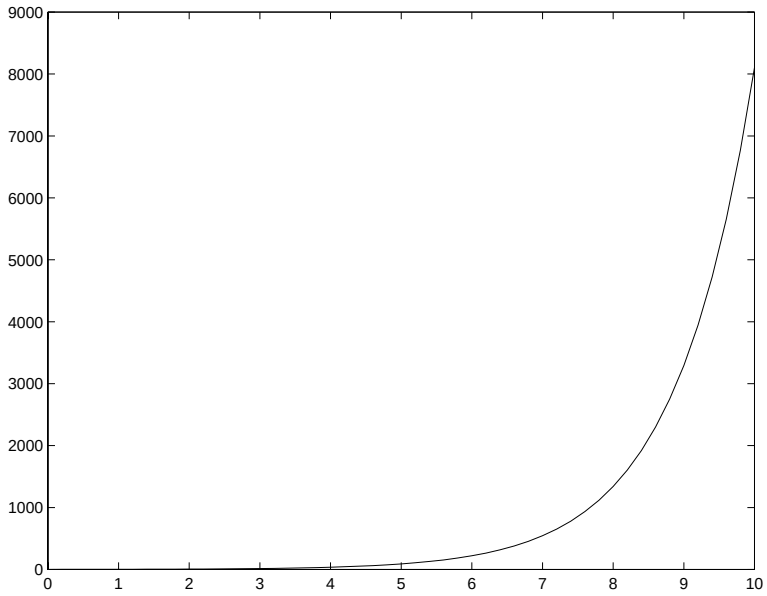
   Note that in a major oversight, Simulink does not provide a mechanism for exporting plots created by the Scope block to word processors. The above figure was generated by the very awkward mechanism of saving the data to the Matlab workspace, plotting it in Matlab, and then exporting the Matlab plot. Regrettably, even that is not simple to do. You have to be sure to export time to the workspace, itself not a totally trivial thing to do. We hope that The MathWorks will improve this situation someday.

3. The result is evidently a decaying exponential function. It must have the form, for all $t \in$ *Reals*$_+$,
   $$z(t) = Ke^{\alpha t}$$
   for some constants K and $alpha$. Since $z(0) = 1$, it must be true that $K = 1$, so
   $$z(t) = e^{\alpha t}$$
   Differentiating this, we find that
   $$\dot{z}(t) = \alpha e^{\alpha t}$$
   from which we can infer that $alpha = a = -0.9$, so
   $$z(t) = e^{at}$$

and

$$\dot{z}(t) = ae^{at} = az(t).$$

4. With the gain set to 0.9 the result looks like this:



The output rapidly blows up. The system is clearly not stable. Once again,

$$z(t) = e^{at}$$

where now $a = 0.9$ This function grows without bound as t gets large.

5. The system is stable for any $a \leq 0$.

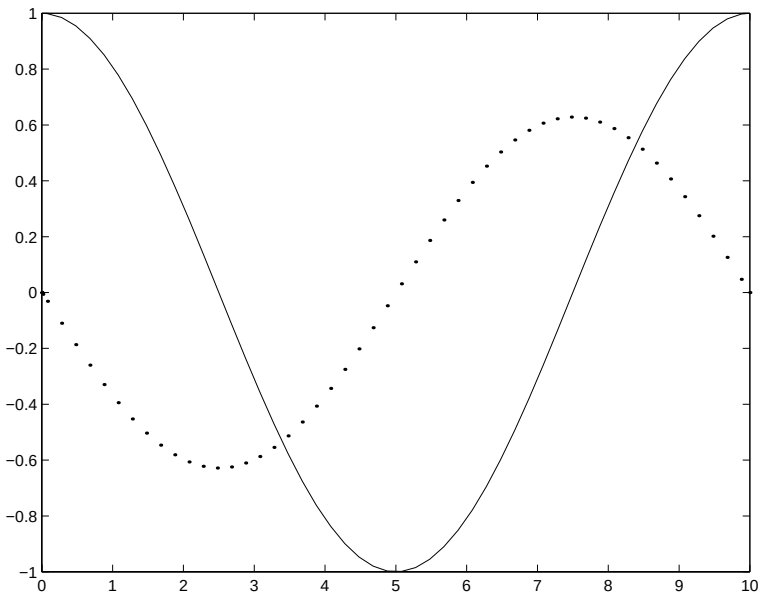
C.6.2 Independent section

1. The A matrix is given by
$$A = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix}.$$

2. The gain of the MatrixGain block is set to

```
[0, 1; -4*pi*pi/100, 0]
```

to make the period 10 seconds. This corresponds to $\omega_0 = 2\pi/10$, which evidently has units of radians per second. The two state variables oscillate with frequency $1/10$ Hz, as shown in the following plot:



3. Letting $\omega_0 = 2\pi \times 440$, we set the gain of the MatrixGain block to

```
[0, 1; -4*pi*pi*440*440, 0]
```

We expect the simulation to complete five cycles in $5/440$ seconds, so we set the simulation to stop at that time. This time, the two state variables have very different amplitudes.

4. The block diagram that generates the tuning fork output is shown in figureC.11. The Matlab command to listen to the output is

```
soundsc(simout(:,1))
```

Note that there is a Simulink block “To Wave Device” that will produce audio output, but it seems more awkward to use than the above method. Nonetheless, it represents a perfectly acceptable solution.

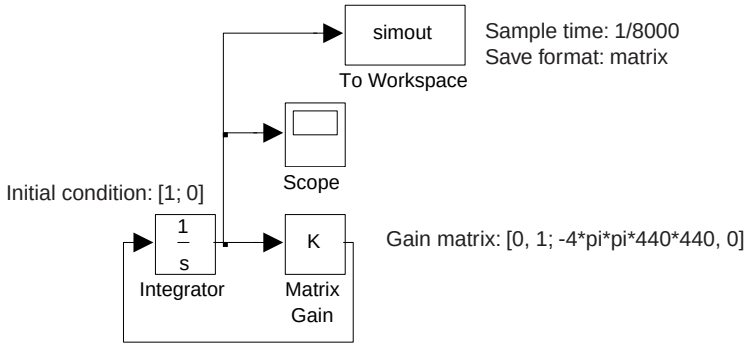


Figure C.11: Block diagram for part 4.

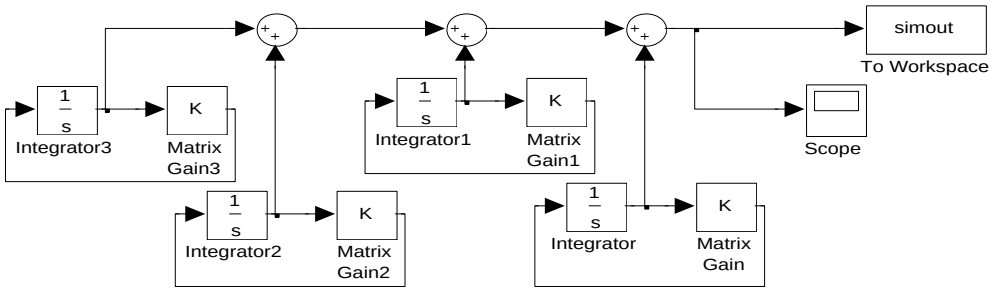


Figure C.12: A block diagram generating a plucked string sound with a fundmental and three harmonics.

5. The sound decays exponentially. Larger negative values of $a_{2,2}$ result in faster decay. Positive values result in exponential growth (the system is unstable). The system is stable if $a_{2,2} < 0$.

6. A block diagram generating a fundamental plus harmonics is show in figure C.12. The four matrix gain blocks have the following parameters, creating a reasonably good plucked-string sound for an A-440:

```
[0, 1; -4*pi*pi*440*440, -5]
[0, 1; -4*pi*pi*440*440*4, -20]
[0, 1; -4*pi*pi*440*440*9, -30]
[0, 1; -4*pi*pi*440*440*16, -40]
```

C.7 Spectrum Solutions

C.7.1 In-lab section

1. The magnitudes of the discrete Fourier series coefficients are plotted in the top plot of figure C.13. This is calculated using the following Matlab code:

```
t = [0:1/8000:1-1/8000];
frequencies = [0:4000];

% First waveform, a sinusoid.
x = sin(2*pi*800*t);
[Ax, Px] = fourierSeries(x);
subplot(3,1,1);
plot(frequencies, Ax);
title('FS coefficients of a sine wave');
ylabel('amplitude');
```

   The horizontal axis is the frequency in Hertz of the sinusoidal component corresponding to each coefficient. Note that only one of the coefficients is non-zero, the one at 800 Hz. It indicates an amplitude of 1 for that component.

2. The chirp is generated and plotted by the following code, which assumes `t` and `frequencies` have been defined as above:

```
% Second waveform, a chirp.
y = sin(2*pi*800*(t.*t));
[Ay, Py] = fourierSeries(y);
subplot(3,1,2);
plot(frequencies, Ay);
title('FS coefficients of a chirp');
ylabel('amplitude');
```

   The magnitudes of the discrete Fourier series coefficients are plotted in the middle plot of figure C.13. That plot shows frequency components ranging from 0 to 1600 Hz, which is exactly the range of instantaneous frequencies predicted by

$$\omega(t) = \frac{d}{dt} 2\pi 800 t^2 = 4\pi 800 t$$

   (divide by 2π to get frequencies in Hz). The odd ringing near 1600 Hz is difficult to explain. It is, in fact, a sort of Gibbs' phenomenon.

3. The reverse chirp is calculated and plotted using:

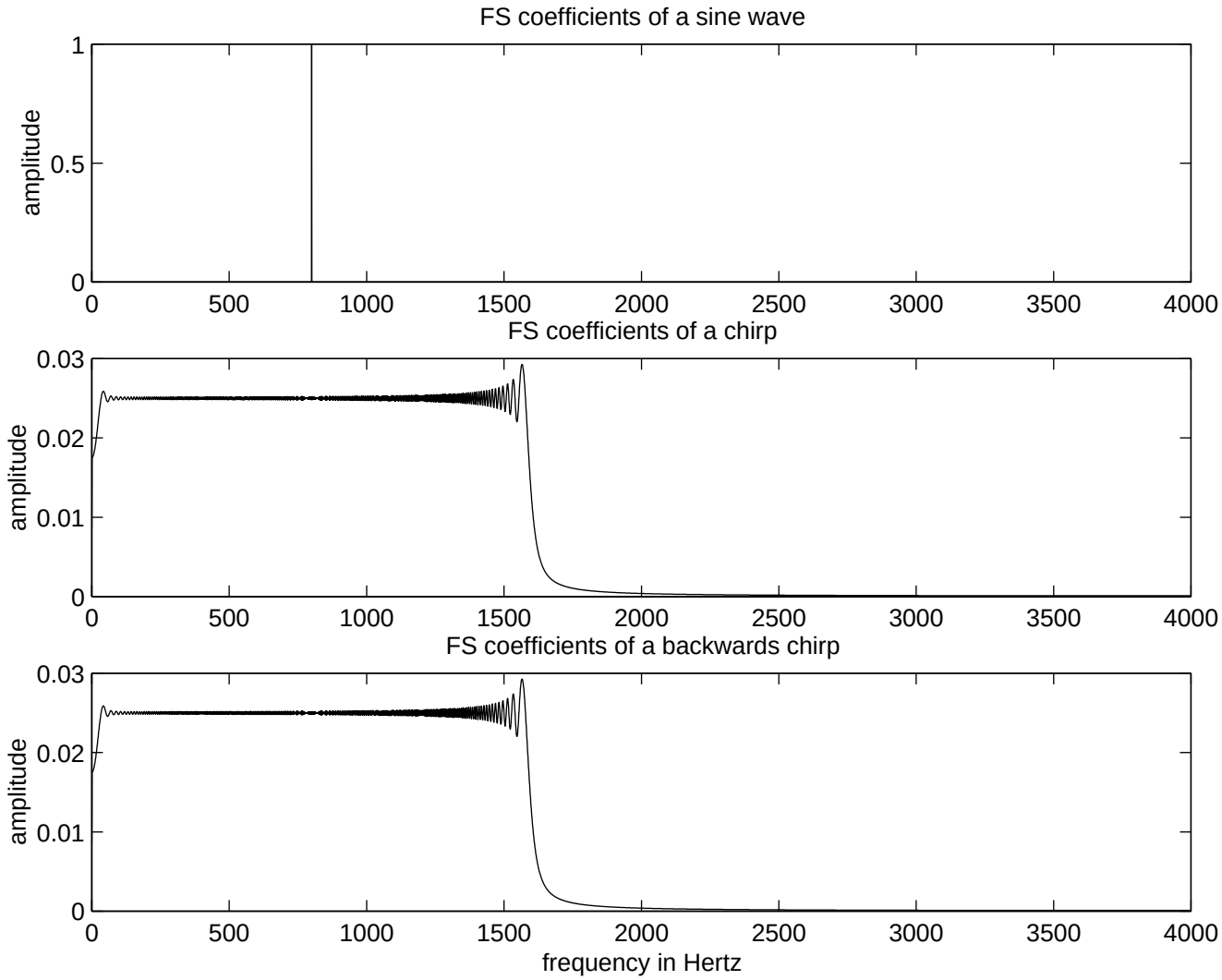


Figure C.13: Discrete Fourier series coefficients of a sine, chirp, and reverse chirp.

```
% Third waveform, a reverse chirp.
z = y(8000:-1:1);
[Az, Pz] = fourierSeries(z);
subplot(3,1,3);
plot(frequencies, Az);
title('FS coefficients of a backwards chirp');
ylabel('amplitude');
xlabel('frequency in Hertz');
```

This is shown in the bottom plot of figure C.13. It is evidently identical to the discrete Fourier series coefficients of the chirp, although it certainly does not sound the same. This suggests that the differences in the two waveforms must be entirely in the phases of their sinusoidal components, which we are not plotting. Moreover, these phase differences are large enough to be perceptible as time shifting of the signal. In fact, this signal is a reverse chirp, where the highest frequency appears first, and the lowest last.

4. In the given plot, the first set of DFS coefficients shows most of the frequency content near zero. Each successive plot shows the frequency content at higher frequencies. This corresponds well with the psychoacoustic perception of the sound, which is that of a sinusoid shifting in frequency from 0 to 1600 Hz.

   The plot for the reverse chirp is shown in figure C.14. That plot shows a spectrum that starts with most of the frequency content at 1600 Hz. It then shifts down to zero. The sound is similar to the chirp, but going from high to low frequencies rather than low to high. The Matlab code for generating the sets of DFS coefficients of the reverse chirp is:

   ```
   waterfallSpectrogram(z, 8000, 400, 30);
   ```

5. The spectrogram for the reverse chirp is generated by the following commands

   ```
   specgram(z,512,8000);
   title('spectrogram of a reverse chirp');
   ```

   The result is shown in figure C.15.

6. The textttttrain.au file yields the spectrogram and plot shown in figure C.16.

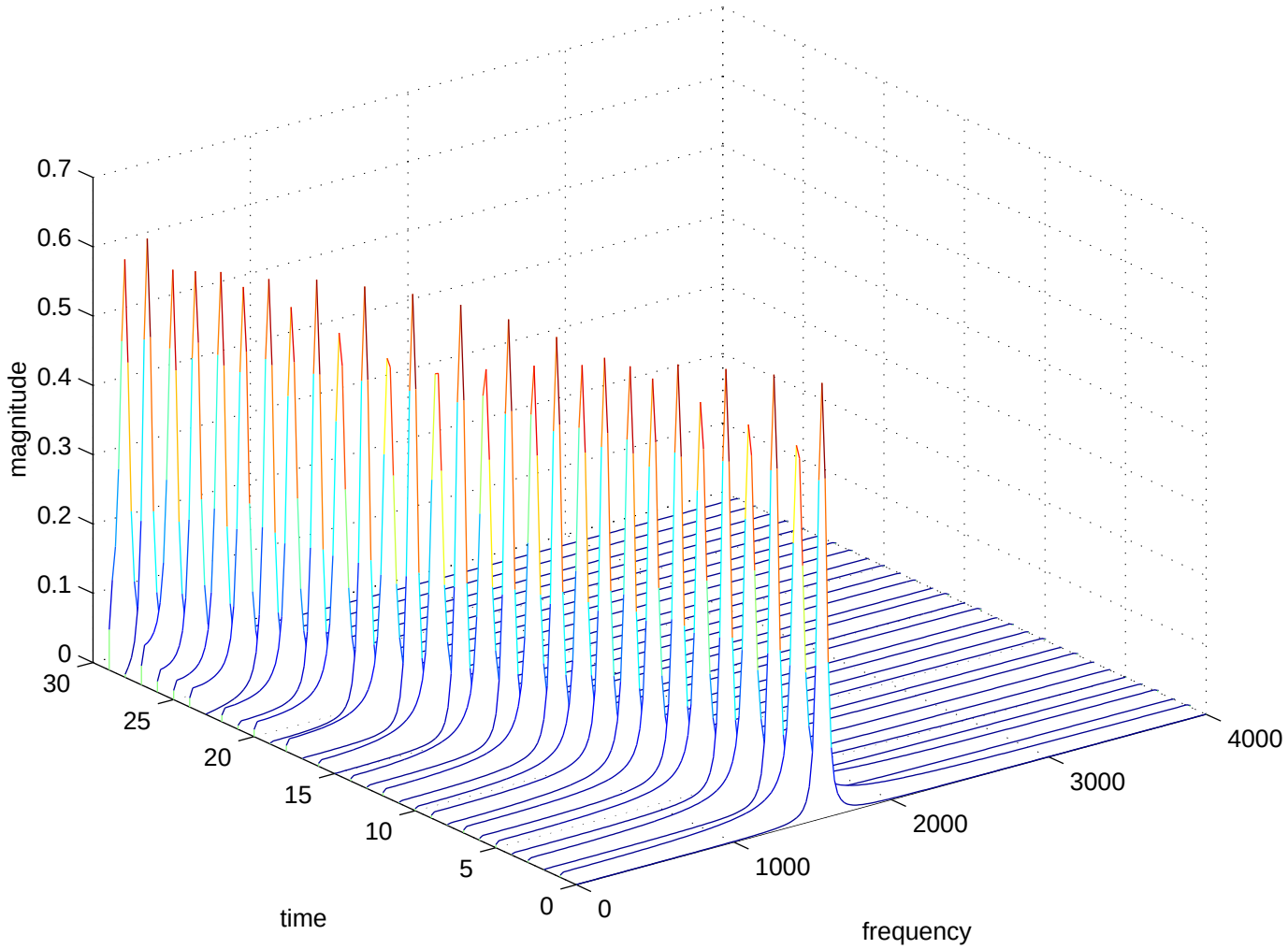


Figure C.14: Time varying discrete Fourier series analysis of a reverse chirp.

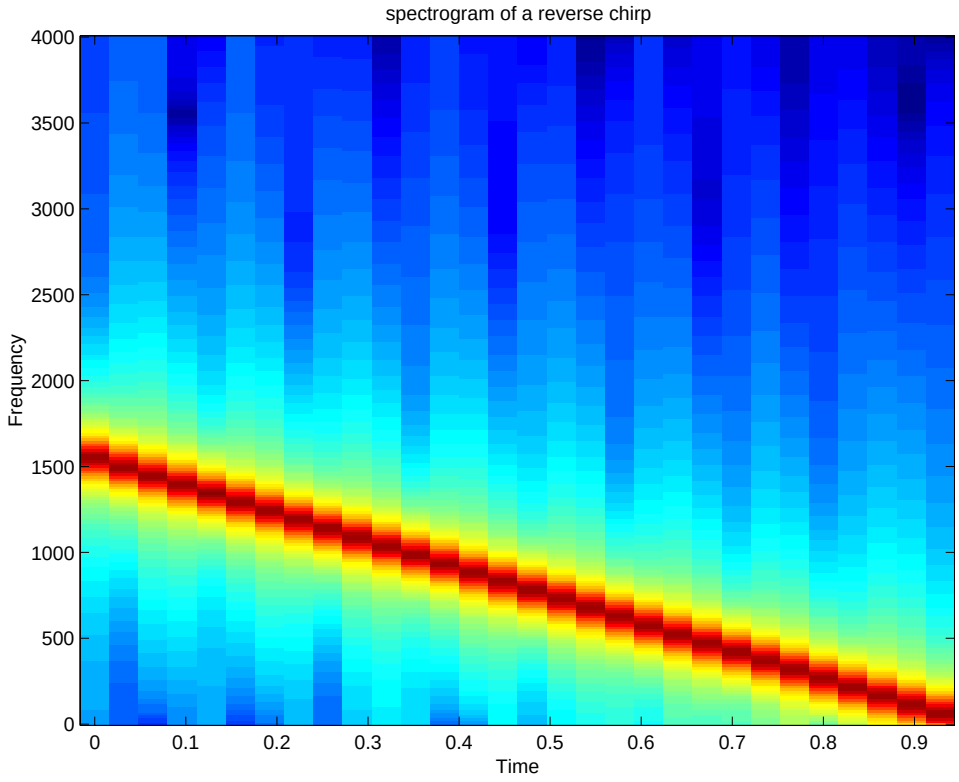


Figure C.15: Spectrogram of the time reversed chirp.

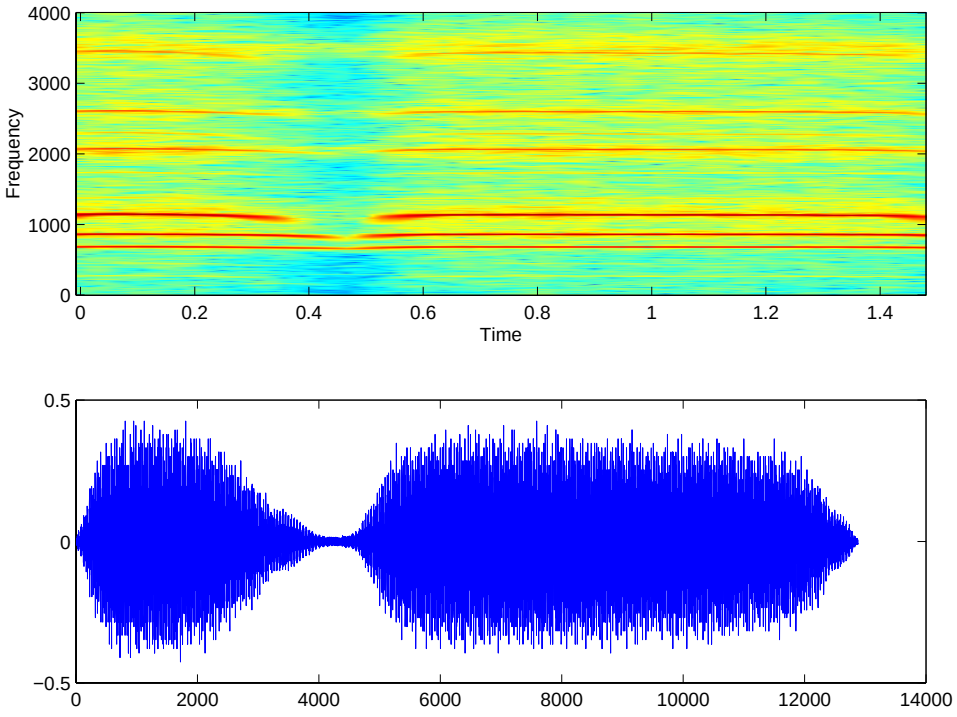


Figure C.16: Spectrogram and waveform for a train whistle.

C.7.2 Independent section

1. One realization of the reconstruction function looks like this:

```
function x = reconstruct(magnitude, phase)
% RECONSTRUCT - Given a vector of magnitudes and a vector
% of phases, construct a signal that has these magnitudes
% and phases as its discrete Fourier series coefficients.
% The arguments are assumed to have odd length, p/2 + 1,
% and the returned vector will have length p.

p = 2*(length(magnitude)-1);
n = 1:p;
x = zeros(1, p);
for k=1:length(magnitude)
    frequency = (k-1)/p;
    x = x + magnitude(k)*cos(2*pi*frequency*(n-1) + phase(k));
end
```

   Notice that we can avoid the for loop, but only at the expense of using a rather large amount of memory to store a large array (8000 by 4000 in this case).

   The following code calculates the discrete Fourier series coefficients reconstructs the original from them, and plots the error in the reconstruction. The result is shown in figureC.17. Notice that the error is mostly smaller than 10^{-12}.

```
t = [0:1/8000:1-1/8000];
y = sin(2*pi*800*(t.*t));
[A, phi] = fourierSeries(y);
yp = reconstruct(A, phi);
error = y - yp;
plot(error);
title('error in reconstruction');
```

2. From results in the specified box,

$$\begin{aligned} 2\cos(\omega_c t)\cos(\omega_\Delta t) &= (e^{i\omega_c t} + e^{-i\omega_c t})(e^{i\omega_\Delta t} + e^{-i\omega_\Delta t})/2 \\ &= (e^{i(\omega_c+\omega_\Delta)t} + e^{-i(\omega_c+\omega_\Delta)t} + e^{i(\omega_c-\omega_\Delta)t} + e^{-i(\omega_c-\omega_\Delta)t})/2 \\ &= \cos((\omega_c + \omega_\Delta)t) + \cos((\omega_c - \omega_\Delta)t). \end{aligned}$$

   where the last equality follows from Euler's relation.

3. The sum of the two sinusoids is generated by:

```
t = [0:1/8000:1-1/8000];
x = cos(2*pi*790*t) + cos(2*pi*810*t);
soundsc(x);
plot(t(1:800), x(1:800));
```

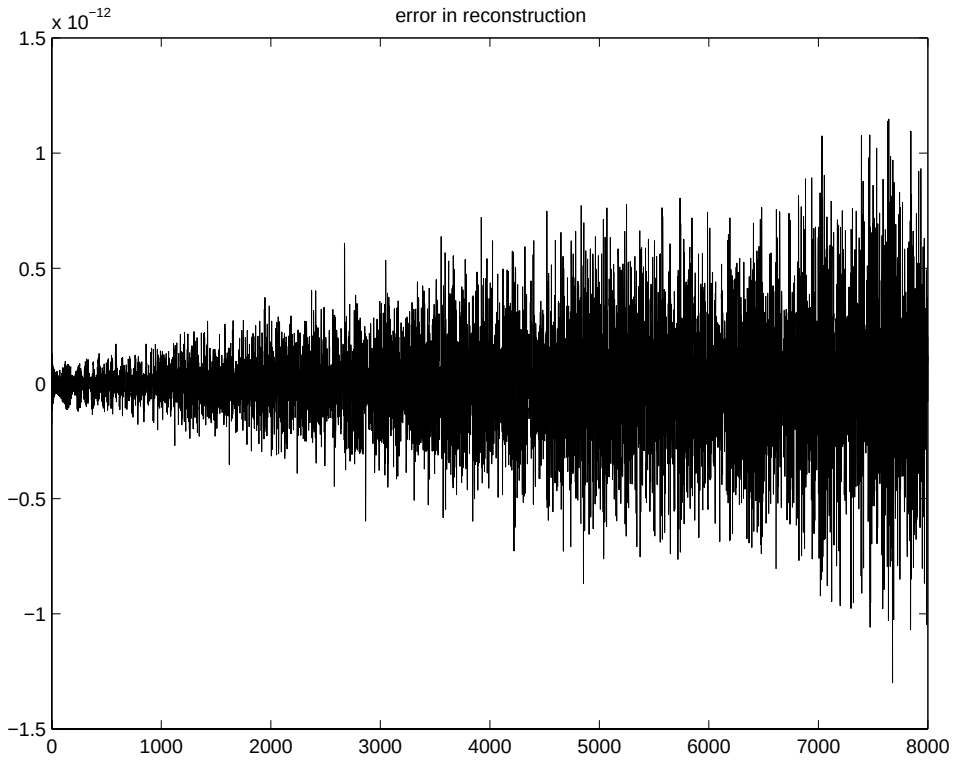


Figure C.17: The reconstruction error from Fourier series coefficients.

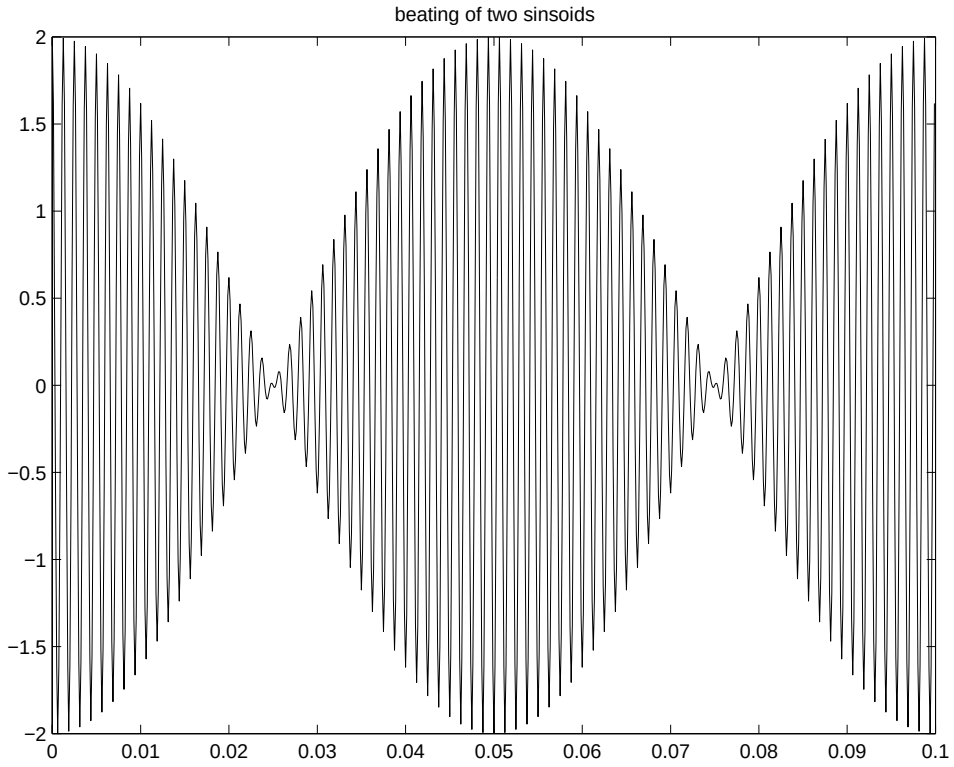


Figure C.18: Beating of two closely spaced sinusoids.

The resulting sound is more like a warble than a pair of tones. The plot of the first 1/10 second in figure C.18 explains why. It shows that the sum results in what looks like a sinusoid of one frequency with an envelope determined by another sinusoid.

To apply the identity in part 2, let $\omega_c = 800 \times 2\pi$ and $\omega_\Delta = 10 \times 2\pi$. Then we see that the sum of two sinusoids at 790 and 810 Hz is equivalent to the product of two sinusoids at 800 and 10 Hz. Indeed, the plot bears this out. The 10 Hz sinusoid imposes the envelope, and we hear the varying amplitude as warble.

4. The period will be the reciprocal of the greatest common divisor of 790 and 810, which is 1/10. This is evident in figure C.18. The fundamental frequency is therefore 10 Hz. The DFS coefficients can be calculated and plotted as follows:

```
[A, phi] = fourierSeries(x);
plot([0:4000], A);
```

The result is shown in figure C.19, which shows two very closely spaced peaks.

The spectrogram can be created as follows:

```
specgram(x,64,8000);
```

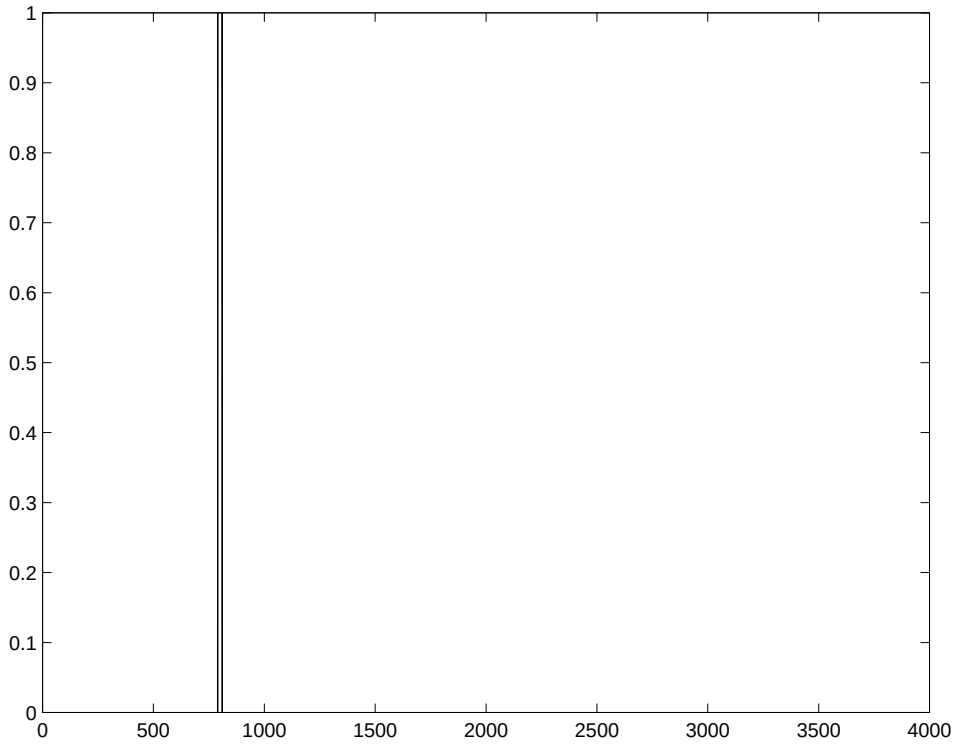


Figure C.19: DFS of two closely spaced sinusoids.

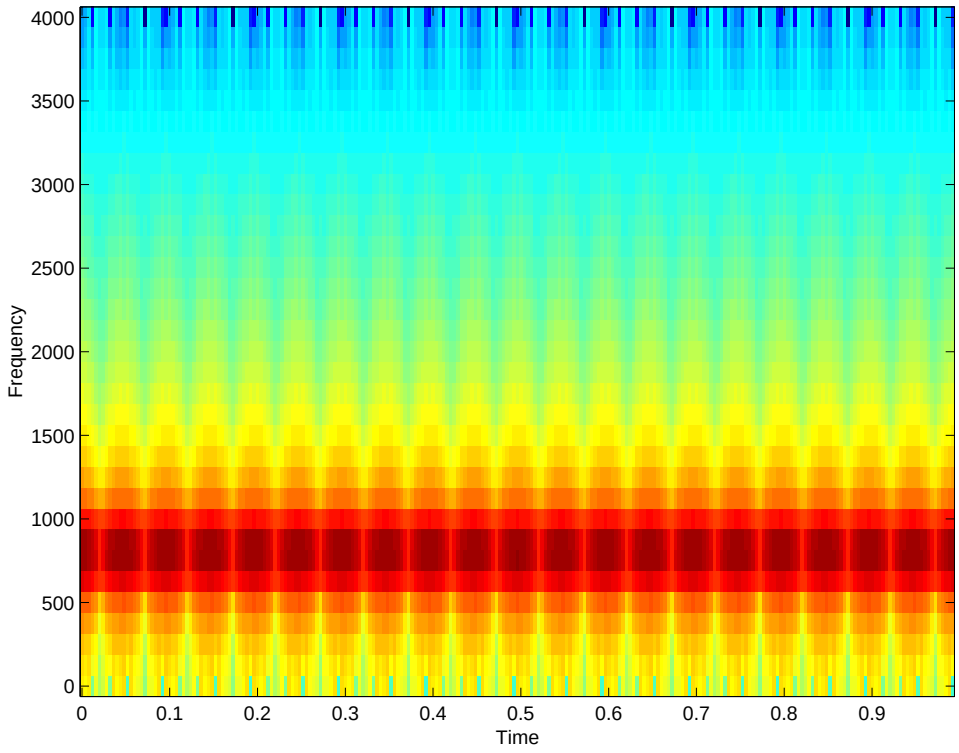


Figure C.20: Spectrogram of two closely spaced sinusoids.

which results in the image shown in figure C.20. This image better reflects perception, because we hear warbling, not two steady tones.

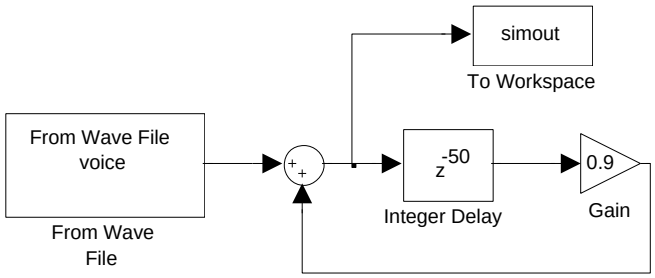


Figure C.21: Solution to part 2.

C.8 Comb Filters Solution

C.8.1 In-lab section

1. S_2 is a system with input y: *Integers* $\rightarrow$ *Reals* and output z: *Integers* $\rightarrow$ *Reals* given by

$$\forall\, n \in \textit{Integers}, \quad z(n) = \alpha y(n - N).$$

   Then

$$y(n) = x(n) + z(n).$$

   The equation describes a system where the output is delayed, scaled, and feb back.

2. This system can be implemented in Simulink as shown in figure C.21. In figure C.21, $\alpha = 0.9$ and and $N = 50$.

   With $N = 2000$, the effect is an echo or reverberation. With $N = 50$, the effect is very different. The sound is similar to that you would hear speaking in round, reverberant chamber, such as a sewer pipe. The feedback in the equation describes reflection of a sound. In a round reverberant chamber one meter in diameter, the sound bounces back and forth with the same amount of delay each time, and each time it is reflected, some energy is lost (hence $\alpha < 1$. The time that it takes to traverse one meter twice is approximately the same as a delay of 50 samples, when the sample rate is 8 kHz, as explained in the hint.

   When $\alpha > 1$, the system becomes unstable. Physically, this would correspond to the sound being amplified on each reflection. A plot of the output with $N = 50$ and $\alpha = 1.1$ is shown in figure C.22, which was created using the following Matlab commands:

```
plot([0:1/8000:4], simout)
xlabel('time'); ylabel('amplitude')
```

   When $\alpha = 0$, the output is identical to the input. When $\alpha = 1$, the system is marginally stable, and the reflections are not attenuated. They echo forever.

3. A plot of the impulse response with $N = 40$ and $\alpha = 0.99$ is shown at the top in figure C.23. A closeup of the first 0.05 seconds is shown at the bottom. These were created using the following Matlab commands:

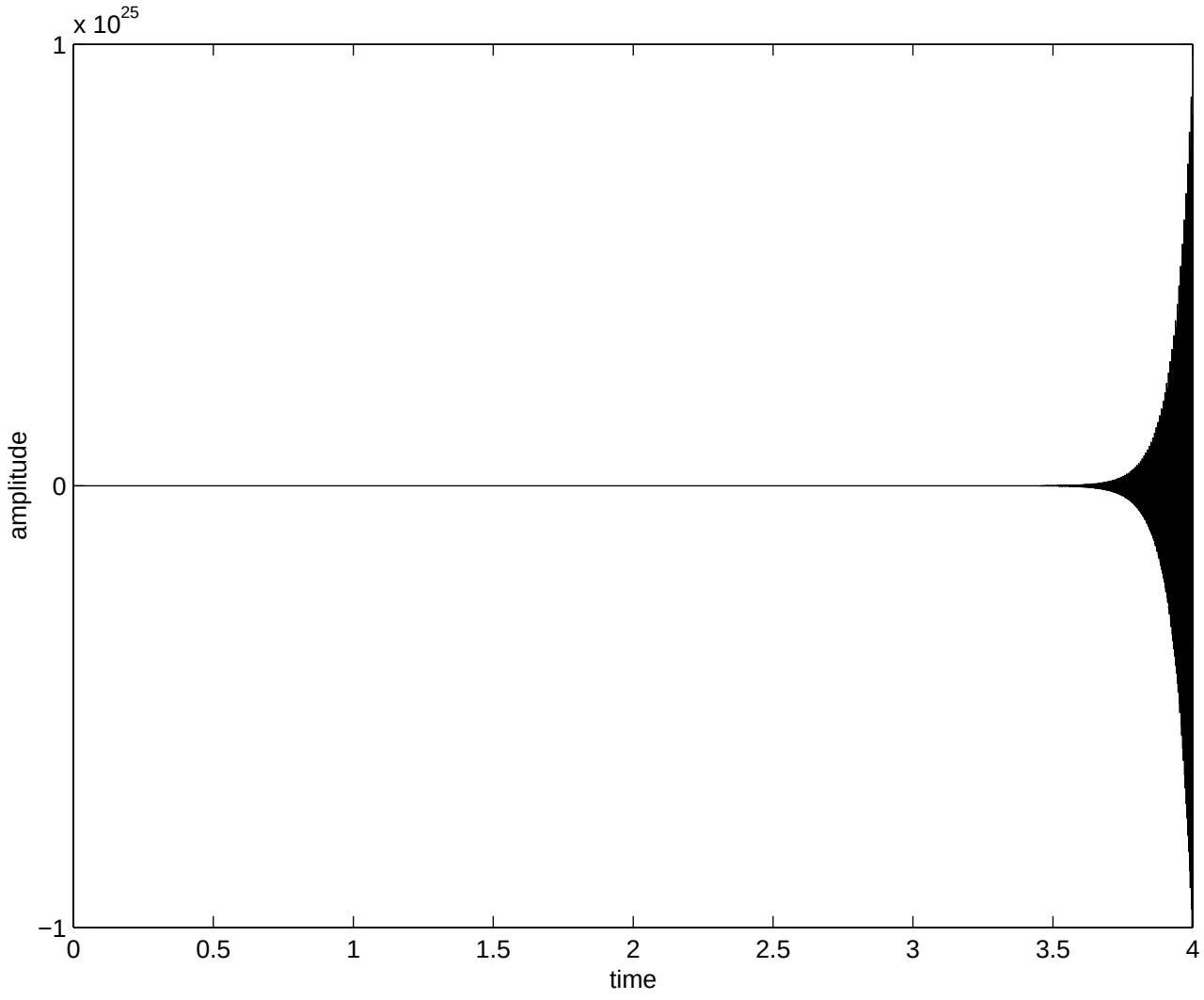


Figure C.22: Output of unstable feedback system.

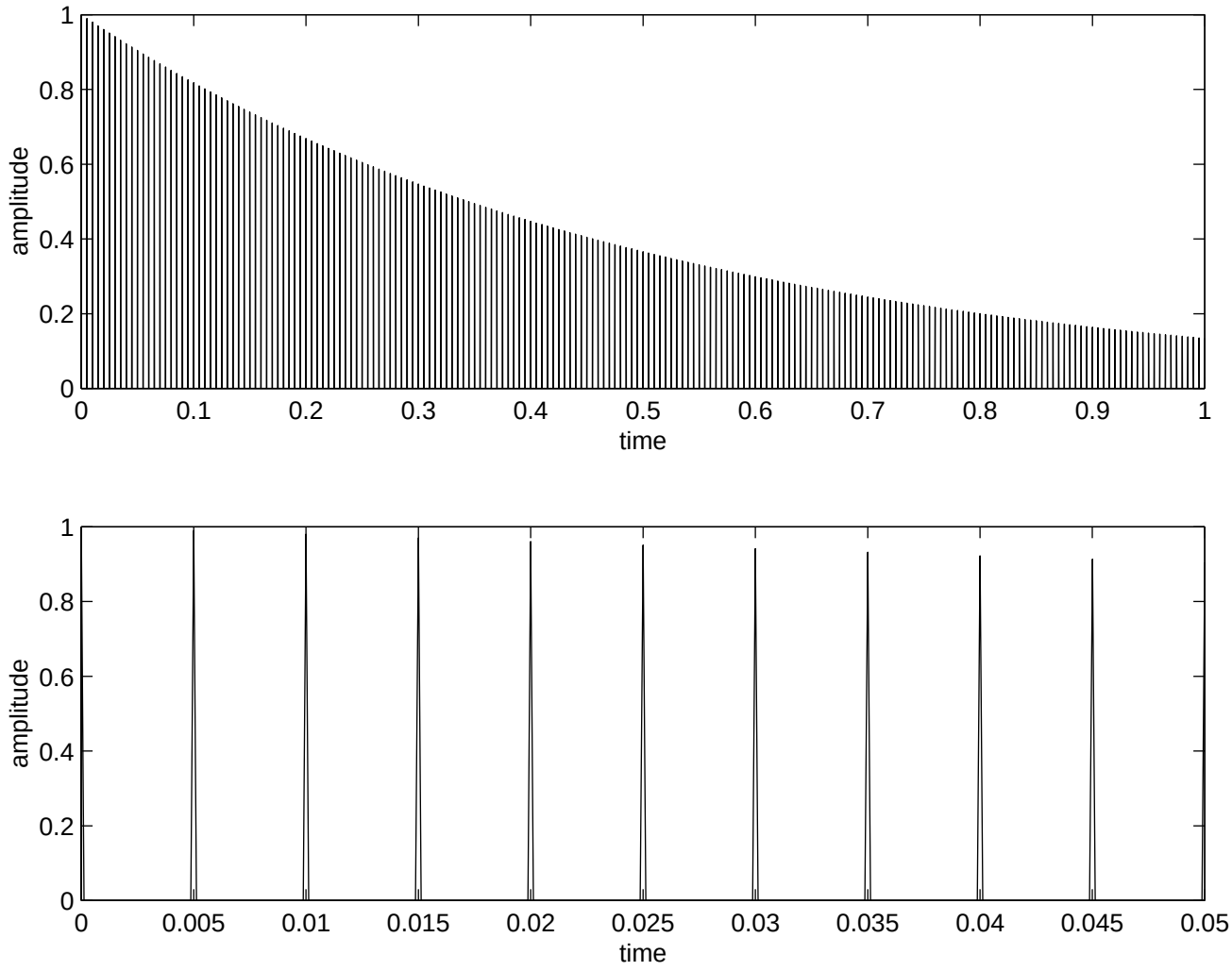


Figure C.23: Impulse response with $N = 40$ and $\alpha = 0.99$.

```
subplot(2,1,1); plot([0:1/8000:1], simout)
xlabel('time'); ylabel('amplitude')
subplot(2,1,2); plot([0:1/8000:0.05], simout(1:401))
axis([0, 0.05, 0, 1]);
xlabel('time'); ylabel('amplitude')
```

Note from the closeup that the impulse response is roughly periodic over short intervals with a period of 0.005 seconds, which corresponds to a fundamental frequency of 200 Hz. This is the tone of the sound we hear, although it is not a musical note (it does not align with any of the frequencies in the musical scale). We will address the problem of getting musical notes in the next lab.

4. The model with random initial values in the delay line is shown in figure C.24. The sound is

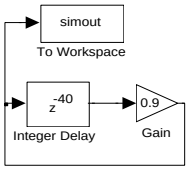


Figure C.24: Comb filter with no input, but random initial values in the delay instead.

indeed richer.

C.8.2 Independent section

1. If the input is

$$x(n) = e^{i\omega n}$$

then the output is

$$y(n) = H(\omega)e^{i\omega n}.$$

We can substitute these into the original equation,

$$y(n) = x(n) + \alpha y(n - N)$$

to get

$$\begin{aligned} H(\omega)e^{i\omega n} &= e^{i\omega n} + \alpha H(\omega)e^{i\omega(n-N)} \\ &= e^{i\omega n}(1 + \alpha H(\omega)e^{-i\omega N}). \end{aligned}$$

Eliminating $e^{i\omega n}$ on both sides we get

$$H(\omega) = 1 + \alpha H(\omega)e^{-i\omega N}.$$

Solving for $H(\omega)$ we get

$$H(\omega) = \frac{1}{1 - \alpha e^{-i\omega N}}.$$

To plot the magnitude of this over 0 to 4 kHz, note that ω varies from 0 to $2\pi \times 4000/8000 = \pi$. We can choose how many samples of the frequency response we wish to plot. Choosing 500, the following Matlab code constructs the plot:

```
omega = 0:pi/500:pi;
alpha = 0.99;
N = 40;
magnitude = abs(1./(1-alpha*exp(-i*omega*N)));
plot(omega, magnitude);
xlabel('frequency');
ylabel('amplitude');
axis([0, pi, 0, 110]);
```

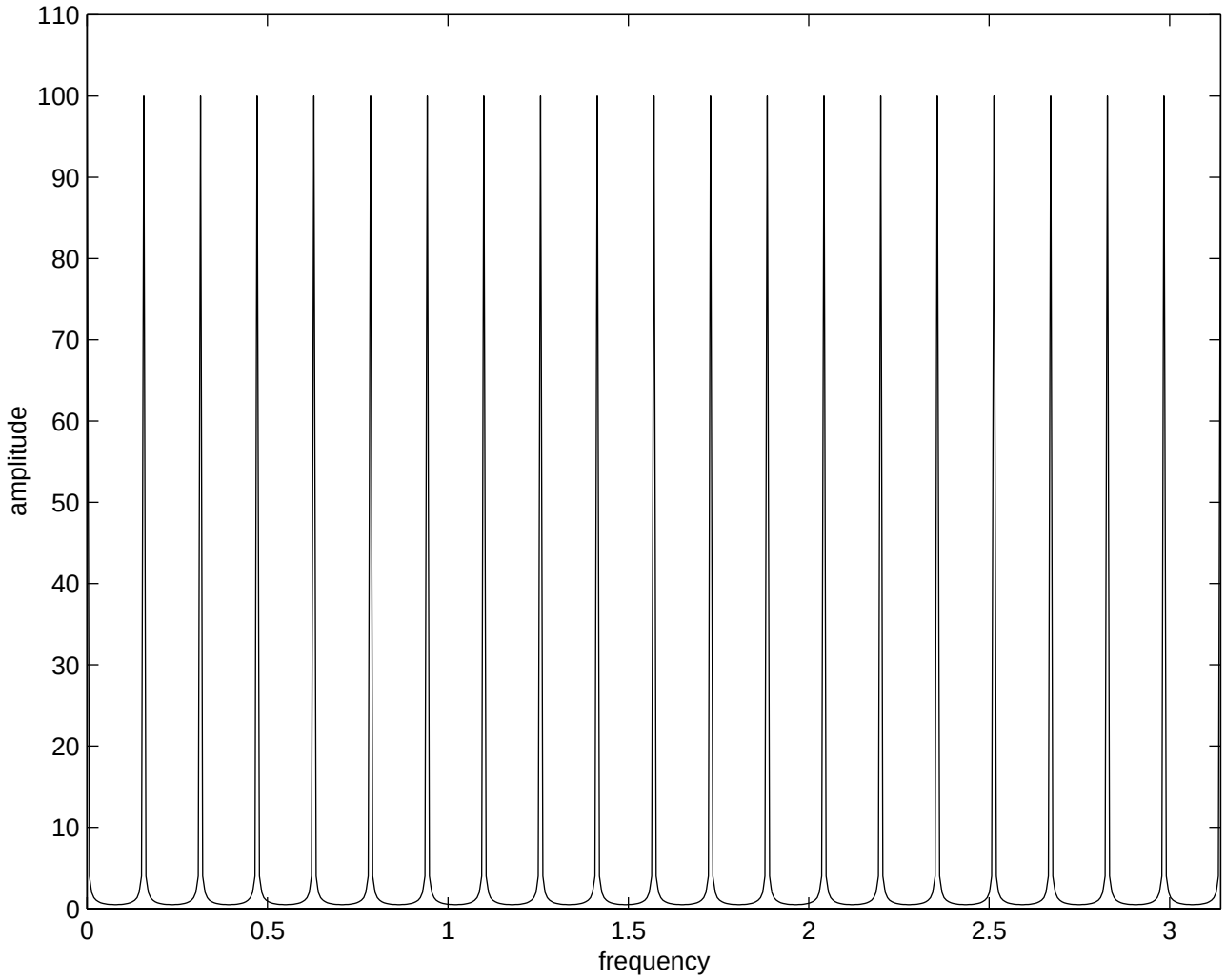


Figure C.25: Frequency response of the comb filter.

This produces the plot shown in figure C.25.

The frequency response indicates that the output will have a fundamental of 200 Hz ($2\pi \times 200/8000 = 0.1571$ radians) plus harmonics at multiples of 200 Hz. This is why the output sounds like a 200 Hz tone, albeit richer than a sinusoidal tone.

C.9 Plucked String Instrument Solutions

C.9.1 In-lab section

1. The moving average is an LTI system, so if the input is $x(n) = e^{j\omega n}$, then the output is $H(\omega)e^{i\omega n}$, where H is the frequency response. We can determine the frequency response using this fact by plugging this input and output into the difference equation, getting

$$H(\omega)e^{i\omega n} = 0.5(e^{i\omega n} + e^{i\omega(n-1)}) = 0.5e^{i\omega n}(1 + e^{-i\omega}).$$

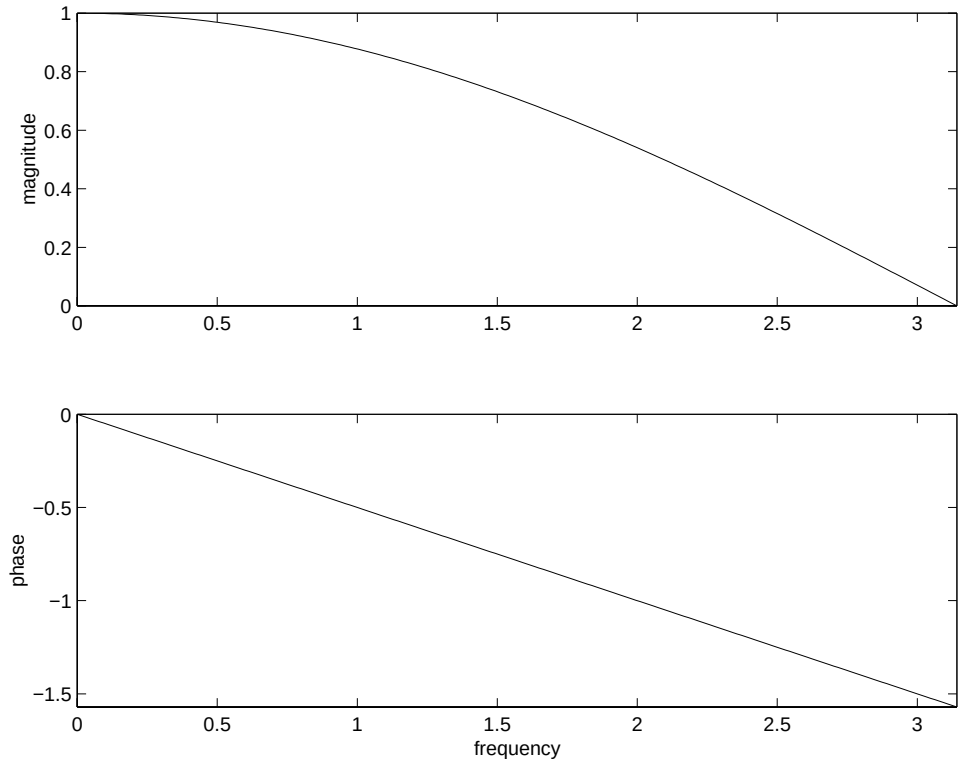
Eliminating $e^{i\omega n}$ on both sides we get

$$H(\omega) = 0.5(1 + e^{-i\omega}).$$

We can plot the magnitude and phase of this using Matlab as follows:

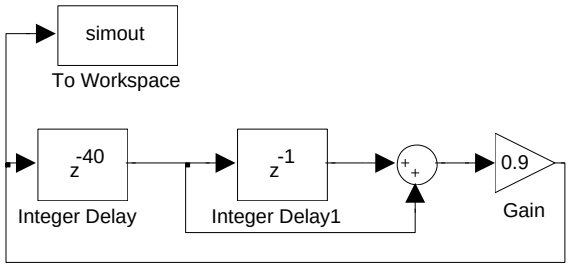
```
omega = 0:pi/200:pi;
H = 0.5*(1+exp(-i*omega));
subplot(2,1,1);
plot(omega, abs(H));
axis([0, pi, 0, 1]);
ylabel('magnitude');
subplot(2,1,2);
plot(omega, angle(H));
axis([0, pi, -pi/2, 0]);
ylabel('phase');
xlabel('frequency');
```

This produces the following plots:



Notice that the phase is indeed linear, and the slope is $-1/2$, which suggests that the delay is $1/2$ sample.

2. A modification of the comb filter model is given below:



The Gain block parameter is set to 0.99*0.5 to account for the 0.5 factor in the lowpass filter. The lowpass filter itself is realized with a unit delay and a summer. As with the comb filter model from the previous lab, the 40 sample delay is initialized with random numbers using

```
randn(1,40)
```

The resulting sound is dramatically more guitar-like.

3. The fundamental frequency with $N = 40$ plus the half-sample delay of the lowpass filter is $8000/40.5 = 197.5$ Hz.

All of the achievable frequencies are of the form $8000/(N + 0.5)$ for integers N. We can compute a range of these in Matlab as follows:

```
>> N = [15:25];
>> 8000./(N+0.5)

ans =

  Columns 1 through 7

  516.1290  484.8485  457.1429  432.4324  410.2564  390.2439  372.0930

  Columns 8 through 11

  355.5556  340.4255  326.5306  313.7255
```

From this, we see that 440 is not achievable. It is flanked by 457 above and 432 below. These are both quite far off, and would not be acceptable for musical purposes.

C.9.2 Independent section

1. The frequency response is given by

$$H(\omega) = 0.5(1 + e^{-i\omega}).$$

We need to find the angle of this as a function of ω. There are a number of ways to do this, but a clever way is to notice that

$$1 = e^{-i\omega/2}e^{i\omega/2}$$

and

$$e^{-i\omega} = e^{-i\omega/2}e^{-i\omega/2}$$

so

$$H(\omega) = 0.5e^{-i\omega/2}(e^{i\omega/2} + e^{-i\omega/2}) = 0.5e^{-i\omega/2}\cos(\omega/2).$$

Over the range of frequencies 0 to π radians/sample, $\cos(\omega/2)$ is non-negative, so the angle is

$$\angle H(\omega) = -\omega/2.$$

This is exactly what we found with the plot, linear phase with a slope of $-1/2$, indicating a delay of 1/2 sample.

2. The difference equation for the comb filter modified with the lowpass filter is

$$\forall\, n \in \textit{Integers}, \quad y(n) = x(n) + 0.5\alpha(y(n-N) + y(n-N-1)).$$

This defines an LTI system, so if the input is $x(n) = e^{j\omega n}$, then the output is $H(\omega)e^{i\omega n}$, where H is the frequency response. We can determine the frequency response using this fact by plugging this input and output into the above to get

$$H(\omega)e^{i\omega n} = e^{i\omega n} + 0.5\alpha(H(\omega)e^{i\omega(n-N)} + H(\omega)e^{i\omega(n-N-1)}).$$

This can be rewritten as

$$H(\omega)e^{i\omega n}(1 - 0.5\alpha e^{-i\omega N} - 0.5\alpha e^{-i\omega(N+1)}) = e^{i\omega n}.$$

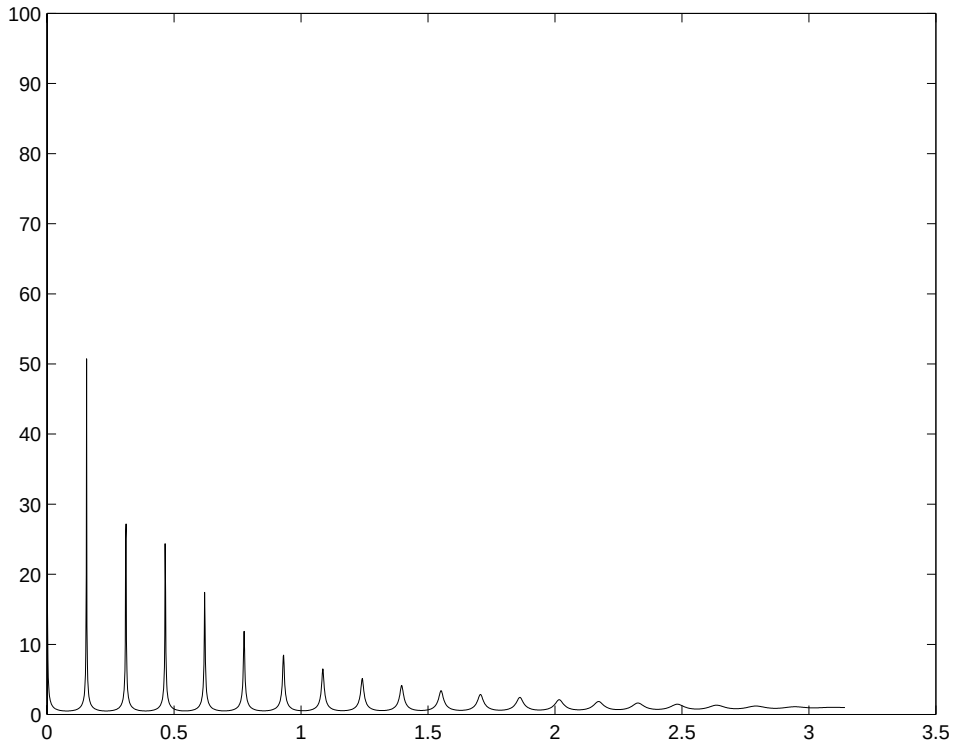
Eliminating $e^{i\omega n}$ and solving for $H(\omega)$ we get

$$H(\omega) = \frac{1}{1 - 0.5\alpha e^{-i\omega N} - 0.5\alpha e^{-i\omega(N+1)}}.$$

To plot the magnitude of this in the range 0 to π we can use the following Matlab commands:

```
omega = 0:pi/2000:pi;
alpha = 0.99;
N = 40;
H = 1./(1 - 0.5*alpha*exp(-i*omega*N) - 0.5*alpha*exp(-i*omega*(N+1)));
plot(omega, abs(H));
```

This results in the plot shown below:



Comparing against the comb filter of the previous lab, we see that the higher frequencies are attenuated relative to the comb filter.

Zooming into this plot, we find that the first peak is at 0.1555 radians/sample. To convert to Hz, we do the following calculation:

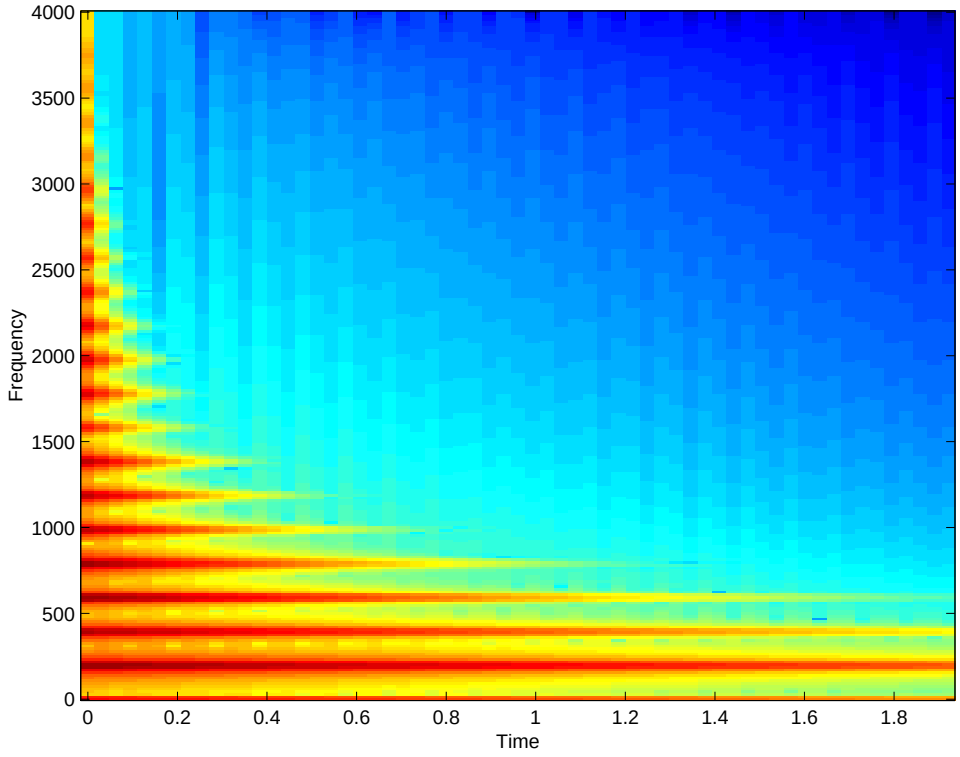
$$0.1555\text{[radians/sample]} \times 8000\text{[samples/second]}/2\pi\text{[radians/cycle]} = 198\text{Hz},$$

very close to the predicted value of 197.5 Hz.

3. The following Matlab command does the job:

```
specgram(simout, 512, 8000)
```

It yields the following image:



This image shows that immediately after starting, the higher harmonics are present and strong, but as the sound progresses, the higher harmonics get weaker.

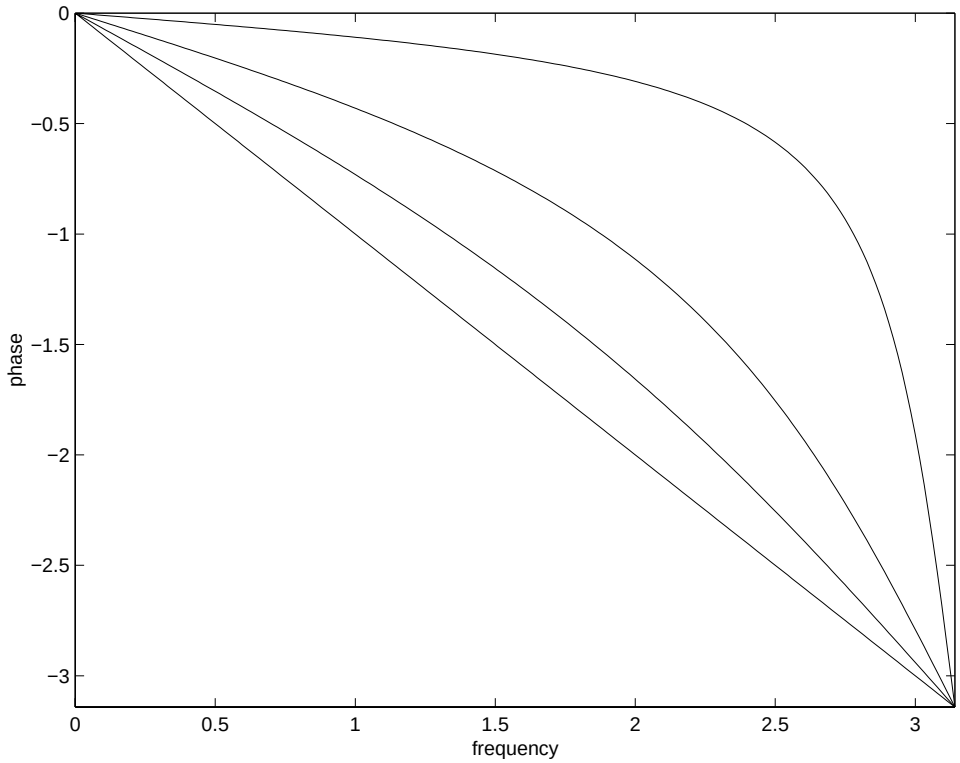
4. Solving for a in terms of d yields

$$a = \frac{1-d}{1+d}.$$

Thus, we can calculate the phase plots using the following Matlab code

```
omega = 0:pi/200:pi;
d = [0.1, 0.4, 0.7, 1.0];
a = (1-d)./(1+d);
for k=1:4
   H(k,:) = (a(k) + exp(-i*omega))./(1 + a(k)*exp(-i*omega));
end
plot(omega, angle(H));
axis([0, pi, -pi, 0]);
xlabel('frequency');
ylabel('phase');
```

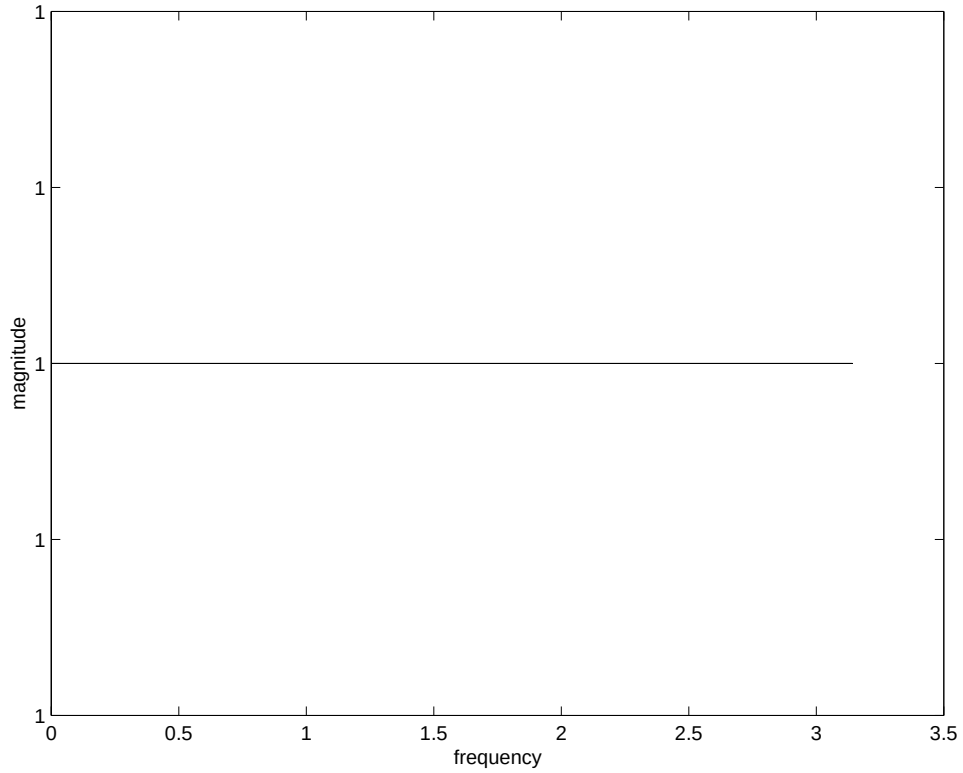
The result is shown below:



This gives the phase response of the allpass filter approximating a delay of 0.1, 0.4, 0.7, and 1.0 samples. Notice that at low frequencies the phase is roughly linear with slope $-d$. For the 1.0 sample delay, the phase is perfectly linear, which should not be surprising since a one sample delay is trivially achieved with discrete-time signals. To get the magnitude response, we just do

```
plot(omega, abs(H));
xlabel('frequency');
ylabel('magnitude');
```

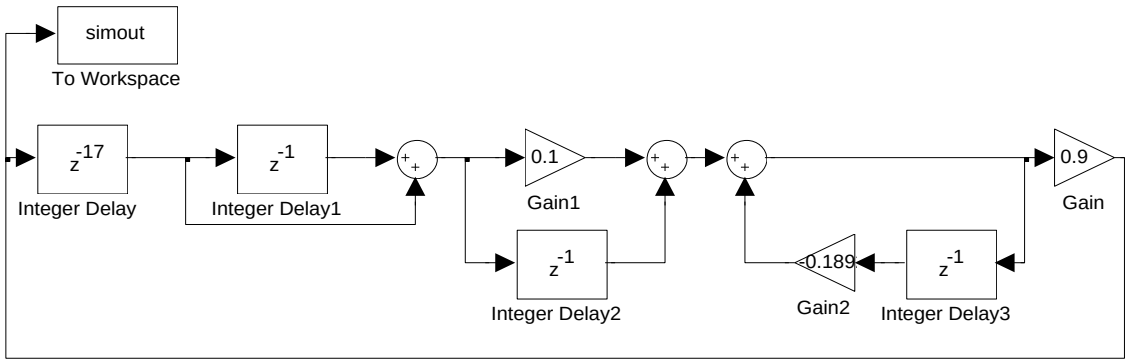
which results in the plot



5. We can rewrite the difference equation for the allpass filter as follows

$$\forall\, n \in \textit{Integers}, \quad y(n) = ax(n) + x(n-1) - ay(n-1)$$

which suggests the implementation shown below:



The delay needed from the allpass filter is a real number $0 < d < 1$ such that

$$8000/(N + 0.5 + d) = 440.$$

Thus

$$N + 0.5 + d = 8000/440 = 18.1818.$$

We choose $N = 17$ so $d = 18.1818 - 17.5 = 0.6818$. From part 4 above we know that

$$a = \frac{1-d}{1+d} = 0.1892.$$

The gain block in the above figure with label "0.1" has the gain parameter set to 0.1892, and the one with label "0.189" has gain -0.1892. Note also that the bulk delay has its parameter set to 17 rather than 40. Also, the feedback gain has been increased to 0.999 rather than 0.99 to get a longer decay of the sound. The result is a good A-440.

C.10 Modulation and Demodulation Solution

C.10.1 In-lab section

1. Note from the DFT definition that

$$\begin{aligned} X'_{k+p} &= \sum_{m=0}^{p-1} x(m)e^{-im(k+p)\omega_0} \\ &= \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}x(m)e^{-imp\omega_0} \\ &= \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}x(m)e^{-im2\pi} \\ &= \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}x(m) \\ &= X'_k \end{aligned}$$

where we have used the facts that $\omega_0 = 2\pi/p$ and $e^{-im2\pi} = 1$ for all integers m.

Note further that

$$X'_{-k} = \sum_{m=0}^{p-1} x(m)e^{imk\omega_0}$$

and

$$(X'_{-k})^* = \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0} = X'_k.$$

2. The following Matlab code gives a correctly labeled plot:

```
t = [0:1/8000:8191/8000];
x = sin(2*pi*100*t + 2*pi*100*(t.*t));
p = length(x);
fs = 8000;
frequencies = [0:fs/p:fs-fs/p];
X = fft(x);
plot(frequencies, abs(X));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The result is shown in figure C.26.

The line

```
frequencies = [0:fs/p:fs-fs/p];
```

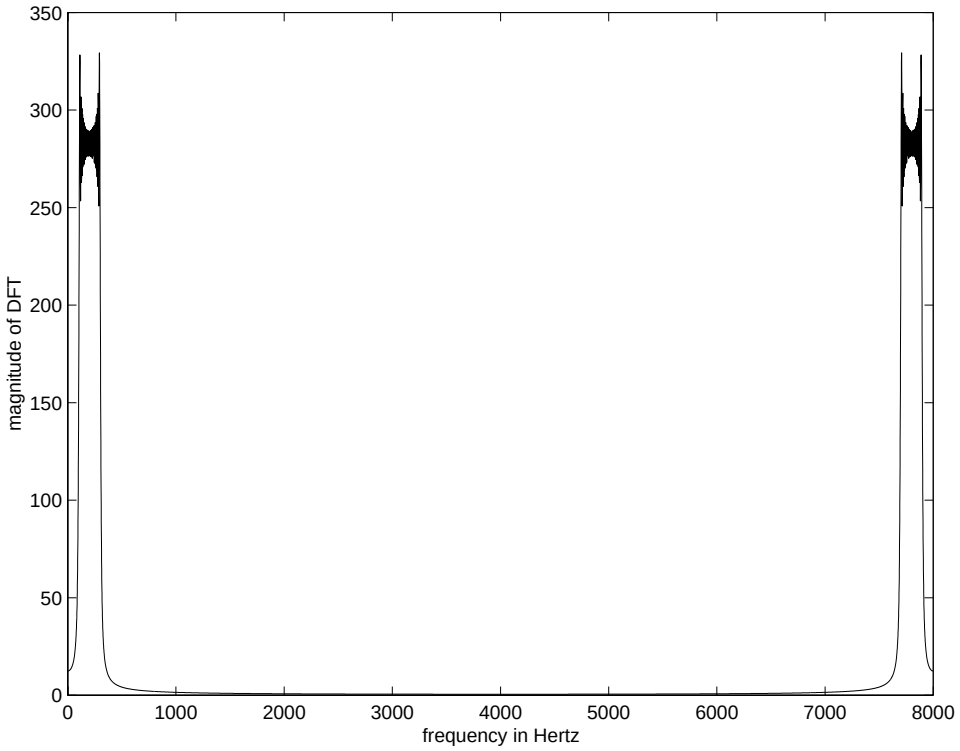


Figure C.26: Magnitude of the DFT of a chirp from 100 to 300 Hz.

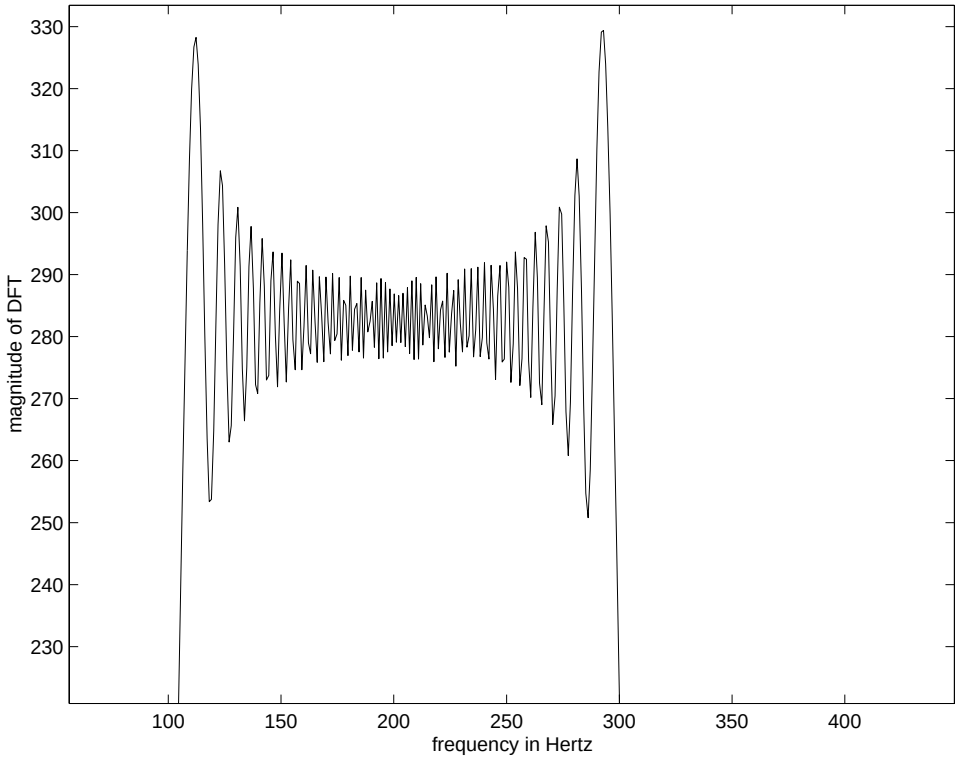


Figure C.27: Magnitude of the DFT of a chirp from 100 to 300 Hz.

bears further examination. It produces a vector with the same length as X, namely p. The elements of the vector are the frequencies in Hertz of each DFT component.

We can zoom into the passband region to verify that the frequency ranges from about 100 Hz to 300 Hz, as shown in C.27. Note that the DFT is not flat in the passband. The reasons for this are subtle, and are beyond the scope of this text.

3. Using the vector X calculated above, all we need to do is exploit the periodicity of the DFT to take the second half of the vector and move it around to the first half. The following Matlab code does the job:

```
Xsymmetric = [X(4097:8192), X(1:4096)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Xsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

Note that $8192/2 = 4096$. The result is shown in figure C.28.

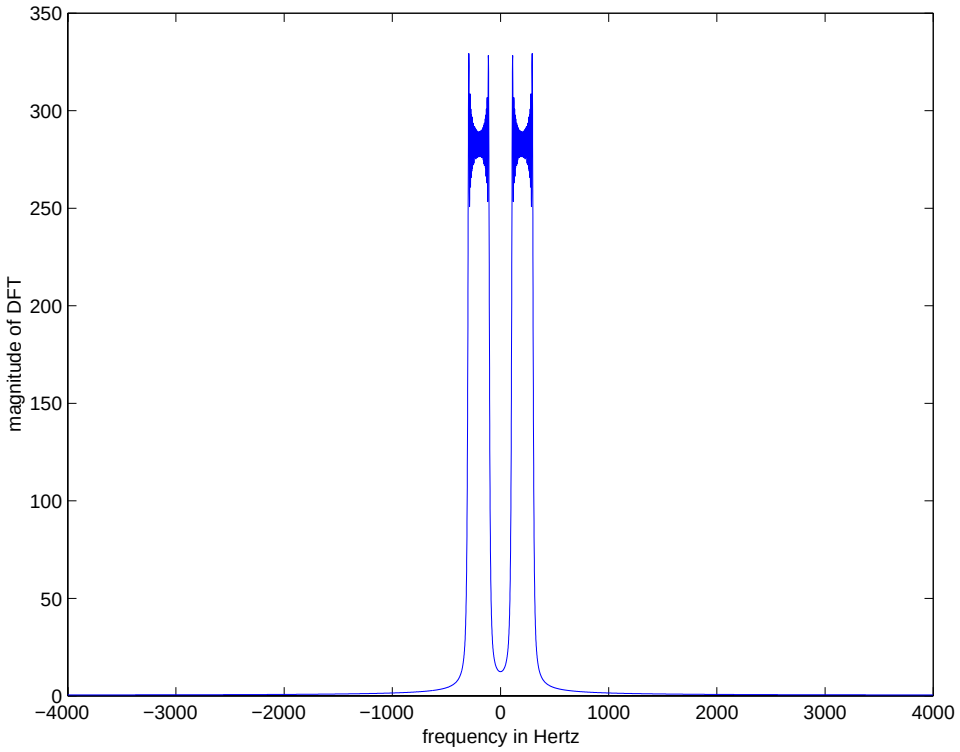


Figure C.28: Magnitude of the DFT of a chirp from 100 to 300 Hz, plotted from -4000 to 4000 Hz.

C.10.2 Independent section

1. The following Matlab code performs the modulation: and plots the DFT magnitude:

```
t = [0:1/8000:8191/8000];
x = sin(2*pi*100*t + 2*pi*100*(t.*t));
carrier = sin(2*pi*1000*t);
y = carrier.*x;
```

   The following Matlab code plots the DFT magnitude:

```
Y = fft(y);
Ysymmetric = [Y(4097:8192), Y(1:4096)];
p = length(y);
fs = 8000;
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Ysymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

   The result is shown in figure C.29. Listening to the sound, we hear a chirp rising from 1100 to 1300 Hz and, simultaneously, another chirp falling from 900 to 700 Hz. The result will get through the specified channel.

2. The following Matlab code, which assumes you have executed the code in the previous part, performs the demodulation:

```
z = carrier.*y;
```

   The following Matlab code plots the DFT magnitude:

```
Z = fft(z);
Zsymmetric = [Z(4097:8192), Z(1:4096)];
p = length(z);
fs = 8000;
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Zsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

   The result is shown in figure C.30. Listening to the sound, we hear the original chirp superimposed with a chirp rising from 2100 to 2300 Hz and, simultaneously, another chirp falling from 1900 to 1700 Hz. This would not be an acceptable demodulation because of the extra chirps.

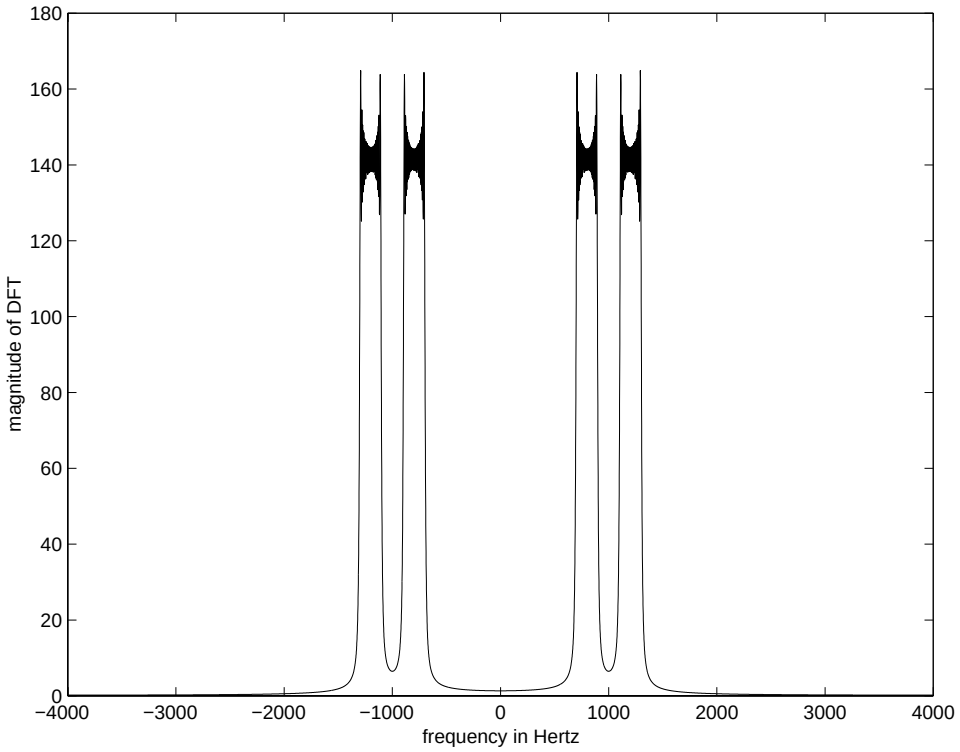


Figure C.29: Magnitude of the DFT of the chirp signal multiplied by a carrier at 1 kHz.

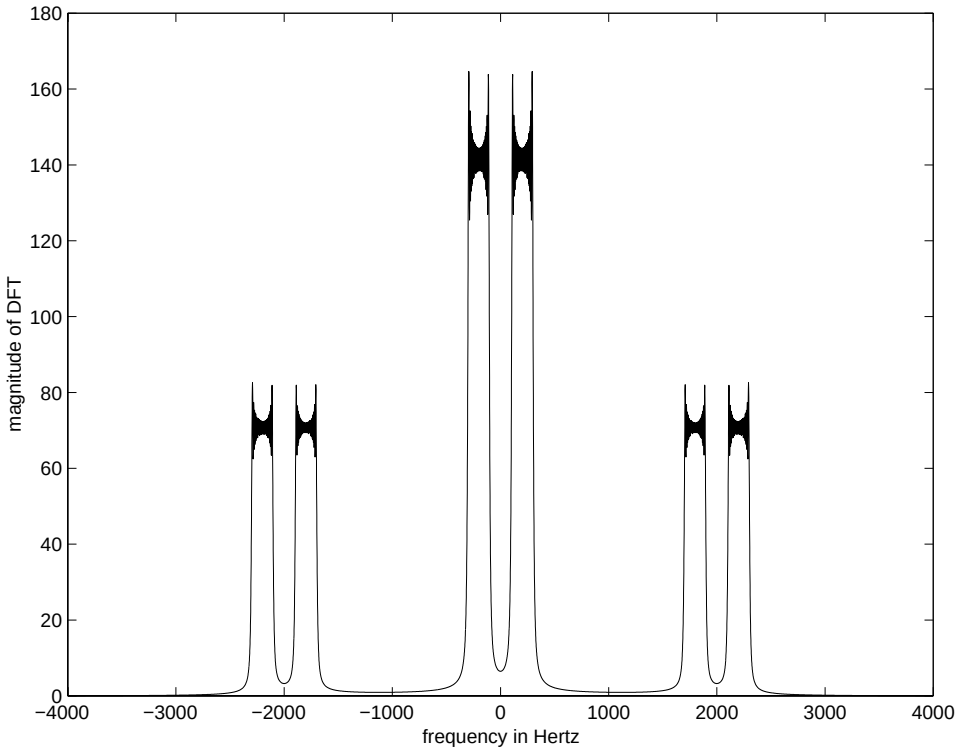


Figure C.30: Magnitude of the DFT of the chirp signal multiplied by a carrier at 1 kHz twice.

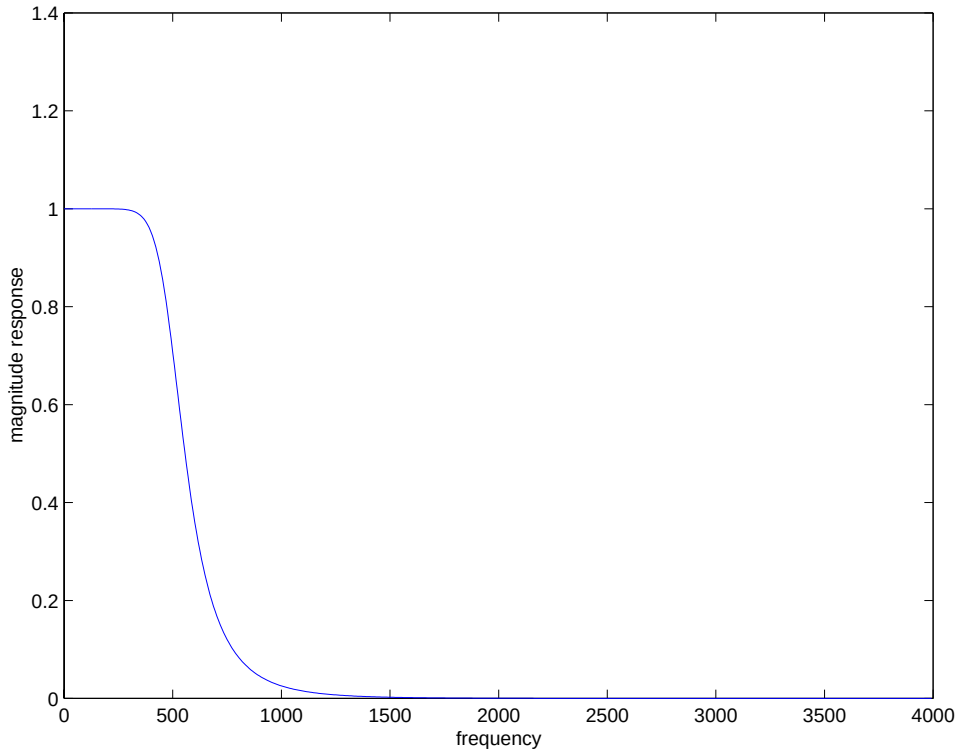


Figure C.31: Magnitude response of a 5-th order Butterworth filter with a cutoff frequency of 500 Hz.

3. We need a filter with a passband that includes the range 100 to 300 Hz, and a stopband that includes the range 1700 to 2300 Hz in order to remove the components centered at twice the carrier frequency. This leaves us with a comfortably wide transition band that can span from 300 Hz to 1700. We could try, for example, a 5-th order Butterworth filter with a cutoff frequency of 500 Hz, which is constructed as follows:

```
[B, A] = butter(5, 0.125);
```

where the second argument follows because $500/4000 = 0.125$. The frequency response of this filter can be plotted using the `freqz` function as follows:

```
[H,W] = freqz(B,A,512);
plot(W*(4000/pi), abs(H));
xlabel('frequency');
ylabel('magnitude response');
```

which yields the plot in figure C.31.

We can filter the demodulated signal and plot its DFT magnitude as follows

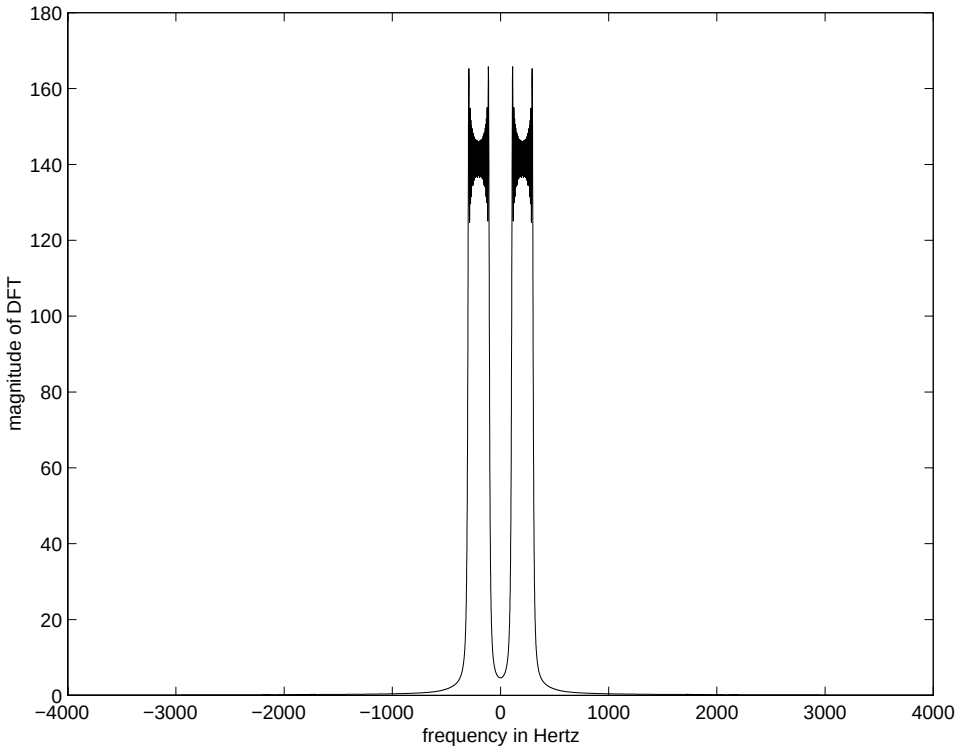


Figure C.32: Magnitude of a demodulated and filtered chirp.

```
w = filter(B, A, z);
W = fft(w);
Wsymmetric = [W(4097:8192), W(1:4096)];
p = length(w);
fs = 8000;
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Wsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The resulting DFT is shown in figure C.32.

If you zoom in sufficiently in the vicinity of 1800 Hz, you can see some trace of the double frequency term. But the result sounds identical to the original chirp.

C.11 Sampling and Aliasing Solution

C.11.1 In-lab section

1. To get a frequency sweep from 0 to 12 kHz in 10 seconds we need to choose f so that $2ft = 12000$ when $t = 10$. Thus, $f = 600$. The following Matlab code generates the chirp and plays it as a sound

```
t = [0:1/8000:10];
y = sin(2*pi*600*(t.*t));
soundsc(y);
```

   The perceived pitch rises from 0 to 4 kHz, then falls back to 0, then rises again to 4 kHz. The reason for this is that the sampled signal has frequency components whose frequencies are ambiguous, modulo $f_s = 8000$. The audio hardware in the computer is choosing to render only frequencies in the range -4 kHz to 4 kHz.

2. To get a frequency sweep from 0 to 2500 Hz in 8192/8000 seconds we need to choose f so that $2ft = 2500$ when $t = 8192/8000$. Thus,

$$f = \frac{2500 \times 8000}{2 \times 8192} = 1221.$$

   The following Matlab code generates this chirp and plays the sound:

```
D = 8192/8000;
t = [0:1/8000:8191/8000];
f = 1221;
y = (1-abs(t-D/2)/(D/2)).*sin(2*pi*f*(t.*t));
soundsc(y);
```

   There are no aliasing artifacts because the frequency remains below the Nyquist frequency.

3. The following Matlab code gives a correctly labeled plot of the DFT of the signal y calculated in the previous part:

```
p = length(y);
fs = 8000;
Y = fft(y);
Ysymmetric = [Y(4097:8192), Y(1:4096)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Ysymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

   The plot is shown in figure C.33. The plot is sensible, in that it shows the triangular shape that determines the relative amounts of each of the frequencies swept.

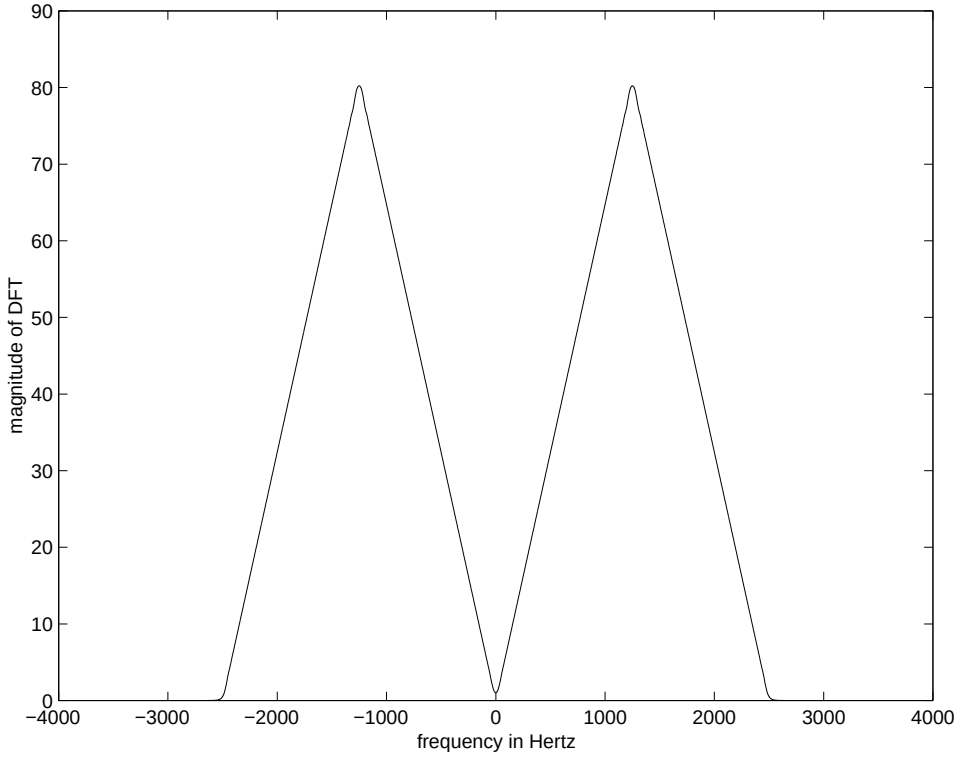


Figure C.33: Magnitude of the DFT of the chirp with a triangular envelope.

4. To get a frequency sweep from 0 to 5000 Hz in 8192/8000 seconds we need to choose f so that $2ft = 5000$ when $t = 8192/8000$. Thus,

$$f = \frac{5000 \times 8000}{2 \times 8192} = 2441.$$

The following Matlab code generates this chirp and plays the sound:

```
D = 8192/8000;
t = [0:1/8000:8191/8000];
f = 2441;
y = (1-abs(t-D/2)/(D/2)).*sin(2*pi*f*(t.*t));
soundsc(y);
```

There is now an aliasing artifact towards the end of the chirp where the frequency drops rather than rises.

The same code as in the previous part can be used to plot the DFT, resulting in the plot shown in figure C.34. Notice that between 3 and 4 kHz there is aliasing distortion. This DFT is the result of overlapping and adding two DFTs that are shifted by 8 kHz in frequency. Since the sweep extends beyond the Nyquist frequency, there is non-zero overlap, resulting in the distortion shown.

5. We recreate the chirp from part 2, which sweeps from 0 to 2500 Hz, as follows:

```
D = 8192/8000;
t = [0:1/8000:8191/8000];
f = 1221;
y = (1-abs(t-D/2)/(D/2)).*sin(2*pi*f*(t.*t));
```

To create the downsampled signal, one (clever) method is

```
w = y([1:4096]*2);
```

The following Matlab code gives a correctly labeled plot of the DFT:

```
p = length(w);
fs = 4000;
W = fft(w);
Wsymmetric = [W(2049:4096), W(1:2048)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Wsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The plot is shown in figure C.35. It shows aliasing distortion because the new sample rate results in a Nyquist frequency that is below the highest frequency terms in the signal.

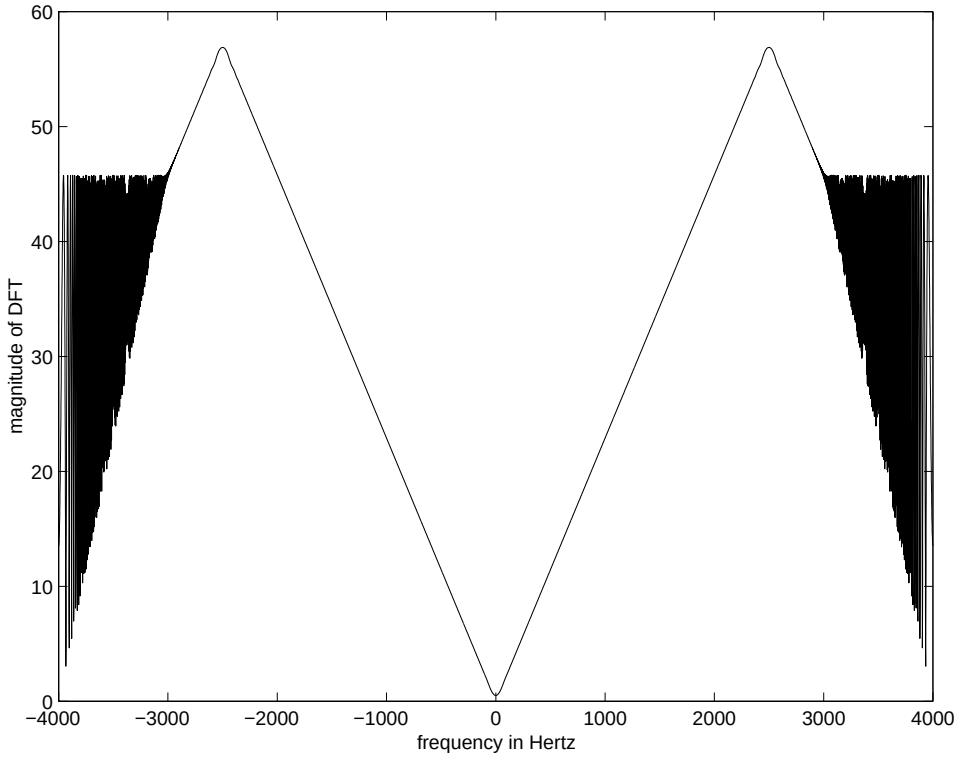


Figure C.34: Magnitude of the DFT of a chirp with a triangular envelope that sweeps beyond the Nyquist frequency.

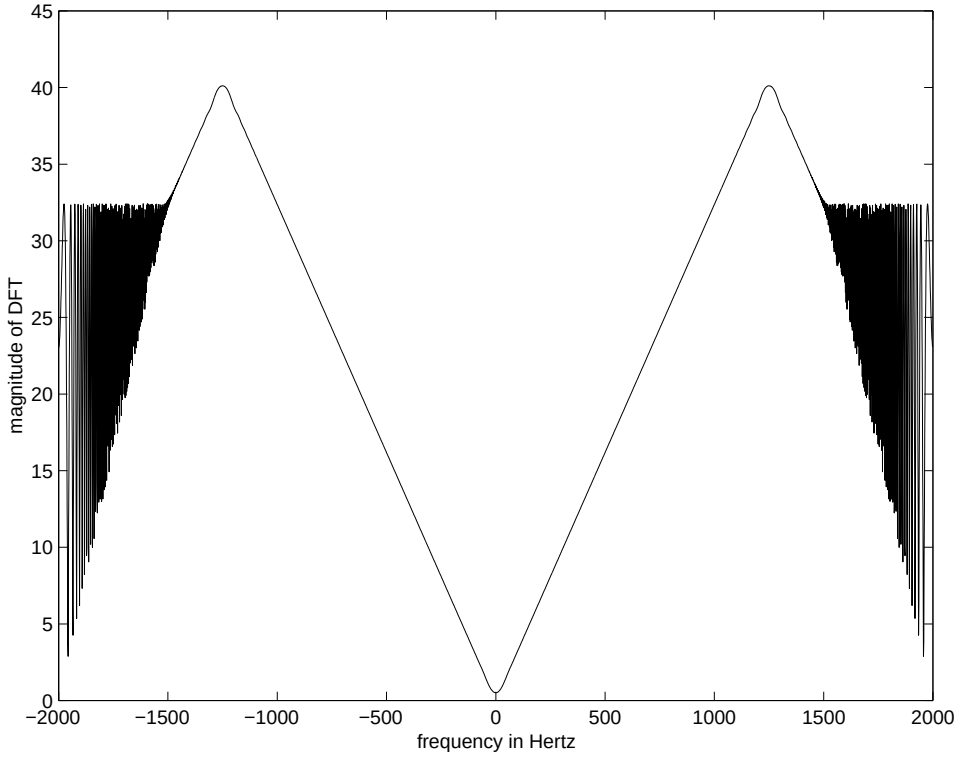


Figure C.35: Magnitude of the DFT of the downsampled chirp.

6. Assuming the same signal y constructed in the previous part, we can construct z as follows:

```
for(k=1:length(y))
  z(2*k-1) = y(k);
  z(2*k) = 0;
end
```

The following Matlab code gives a correctly labeled plot of the DFT:

```
p = length(z);
fs = 16000;
Z = fft(z);
Zsymmetric = [Z(8193:16384), Z(1:8192)];
symmetricFreqs = [-fs/2:fs/p:(fs/2)-(fs/p)];
plot(symmetricFreqs, abs(Zsymmetric));
xlabel('frequency in Hertz');
ylabel('magnitude of DFT');
```

The plot is shown in figure C.36.

7. To get back the original chirp, we could lowpass filter the signal with the following Butterworth filter:

```
[B, A] = butter(10, 1/3);
u = filter(B, A, z);
```

The result sounds just like the original chirp. You can plot its magnitude DFT as above, and it will look perfect. If you use a lower order filter, such as 4 instead of 10, then there will be small residuals in the frequency range 5500 to 8000 Hz.

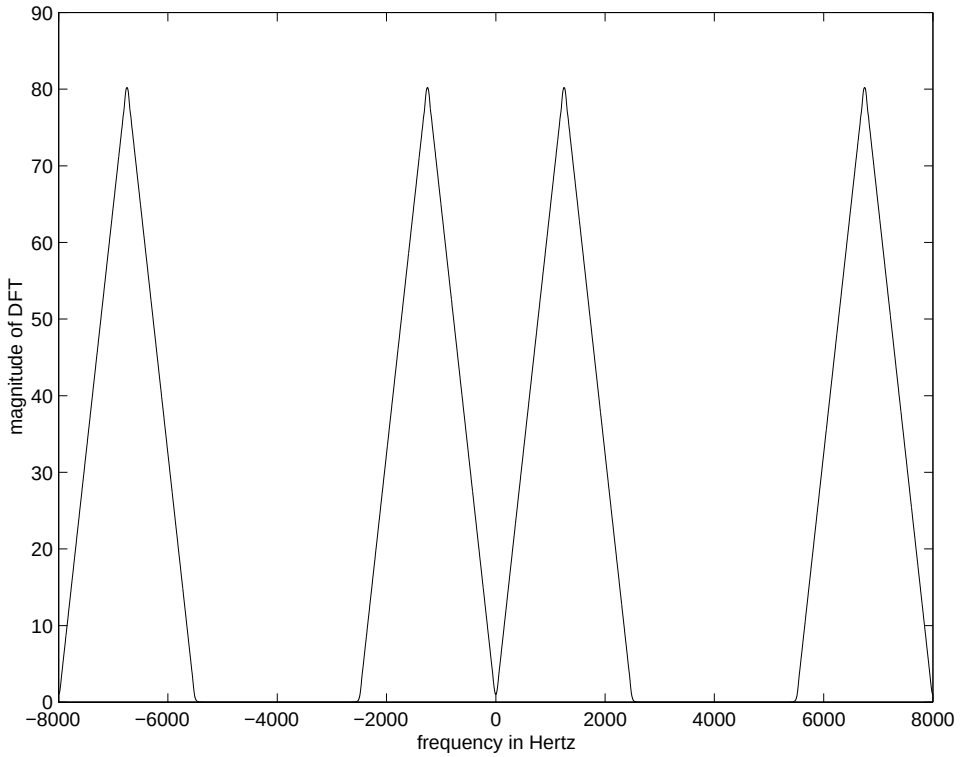


Figure C.36: Magnitude DFT of an upsampled chirp with zero samples inserted.