SEMESTER 2 2018

MARS-242 MISSION CONTROL

EGB242 - ASSIGNMENT 1A

GROUP 3

CALLUM LAING – N8797617

HOANG CHANH PHAM – N9889574

KEVIN DUONG – N9934731

Introduction

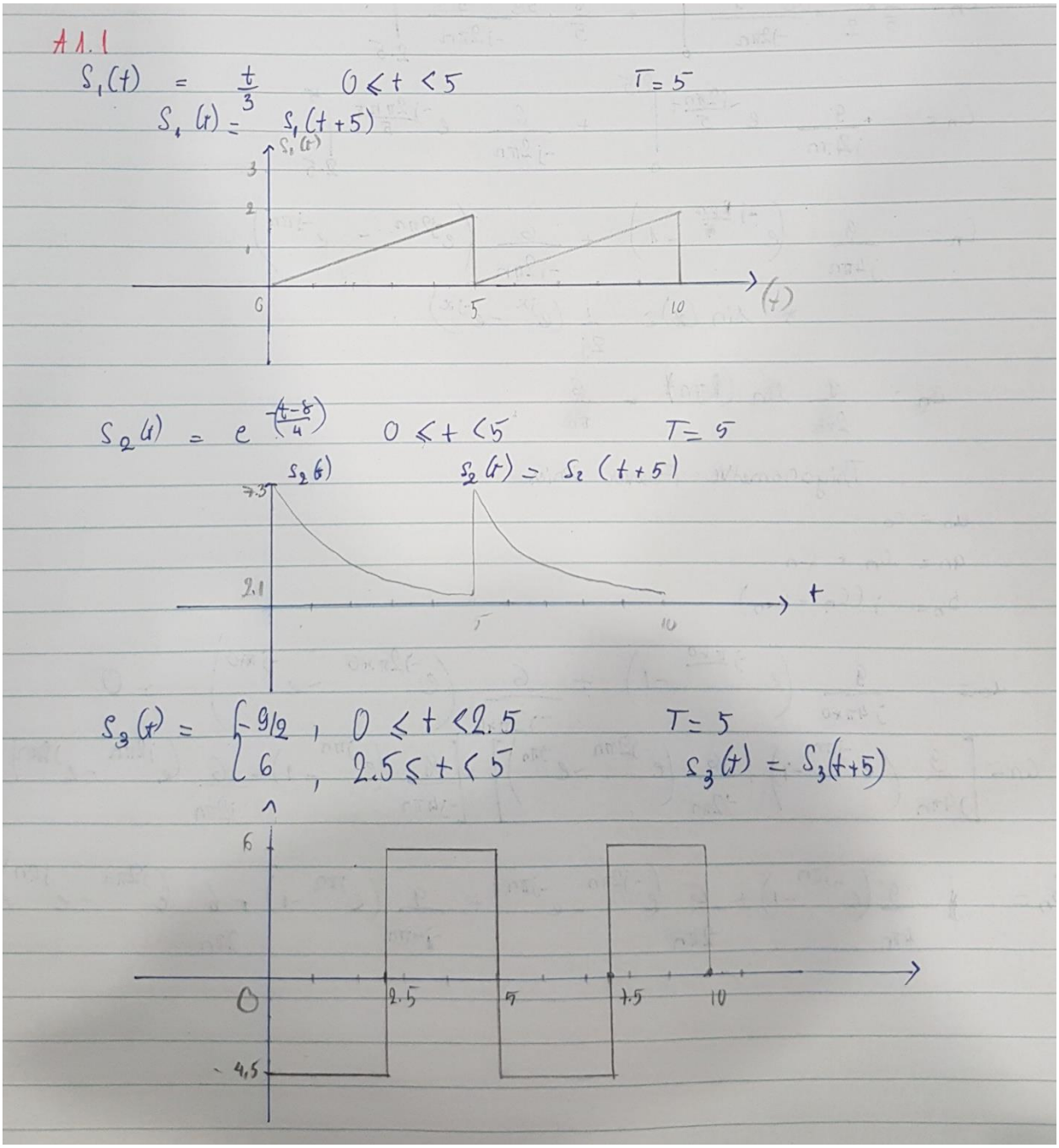
A space agency in Australia called Brisbane-based Australian Space Agency (BASA) has been working on a project to reach to Mars. The engineering team in charge of this mission are looking for capable engineers to work with them, and have set various tasks for potential candidates to demonstrate the knowledge, skills and abilities that are required to join the BASA team. Their overall objective is to ensure the safety of the MARS-242 team of astronauts who are heading to mars in their spaceship, by handling the operation and monitoring the astronauts on-board.

This report discusses three tasks that lead to the investigation of the interference affecting the speech received by the astronauts from their spaceship, assuming the communication disruptions to be periodic noise. The tasks involve problem solving a set of periodic functions through mathematical techniques such as trigonometric and complex exponential fourier series. It also looks at the training exercise BASA has provided - through the use of MATLAB code that needs to be solved, and finally the speech received that needs to be denoised in order to hear the message play out properly.

A1 - BASA Headquarters: Problem Solving

Three test signals: s1(t), s2(t) and s3(t) are defined to be used and solved in all three sections. This first section looks at the following signals and solves them thoroughly with a set of questions given by BASA. The answers are handwritten and deals greatly with trigonometric and complex exponential Fourier series for each of the given signal.

A1.1 - Graphing periodic functions



A1.2 - Trigonometric and complex exponential fourier series of s1(t)

A1.2

$S_1(t) = \frac{t}{A} \quad 0 \le t < 5 \qquad A = 3 \qquad \omega = \frac{2\pi}{5}$

$a_0 = \frac{1}{T}\int_0^T S_1(t)\,dt = \frac{1}{5}\int_0^5 \frac{t}{3}\,dt = \frac{1}{5}\,\frac{t^2}{6}\Big|_0^5 = \frac{5}{6}$

$a_n = \frac{2}{T}\int_0^5 s_1(t)\,dt \cos(n\omega t)\,dt$

$= \frac{2}{15}\int_0^5 t\cos(n\omega t)\,dt$

$= \frac{2}{15}\left[\frac{t\sin n\omega t}{n\omega}\Big|_0^5 - \int_0^5 \frac{\sin n\omega t\,dt}{n\omega}\right]$

$= \frac{2}{15}\left[\frac{t\sin n\omega t}{n\omega} + \frac{1}{n\omega}\,\frac{\cos n\omega t}{n\omega}\right]_0^5$

$= \frac{2}{15}\left[\frac{t\sin\frac{2\pi n}{5}t}{\frac{2\pi n}{5}} + \frac{1}{4n^2\frac{(2\pi)^2}{5^2}}\cos\frac{2\pi n}{5}t\right]_0^5$

$= 0$

$b_n = \frac{2}{T}\int_0^5 S_1(t)\sin(n\omega t)\,dt = \frac{2}{15}\int_0^5 t\sin(n\omega t)\,dt$

$= \frac{2}{15}\left[\frac{t(-\cos(n\omega t))}{n\omega}\Big|_0^5 + \int_0^5 \frac{\cos n\omega t}{n\omega}\,dt\right]$

$= \frac{2}{15}\left[-\frac{t\cos n\omega t}{n\omega} + \frac{\sin n\omega t}{n^2\omega^2}\right]_0^5$

$= \frac{2}{15}\left[-\frac{t\cos\frac{2\pi n}{5}t}{\frac{2\pi n}{5}} + \frac{\sin\frac{2\pi n}{5}t}{\frac{4\pi^2 n^2}{25}}\right]_0^5$

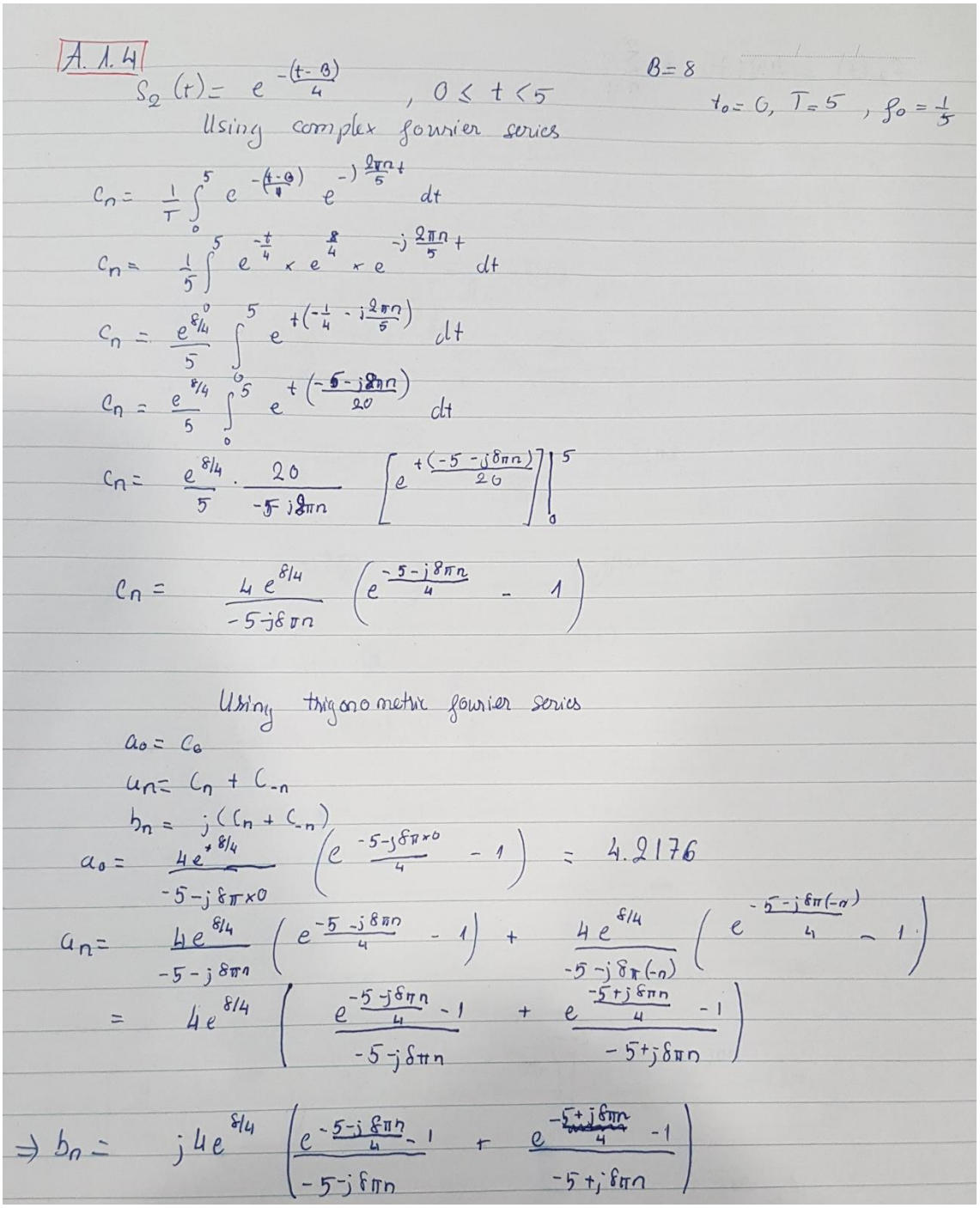
$= \frac{2}{15}\cdot\frac{-5}{2\pi n}\cdot 5 = \frac{-5}{3\pi n}$

$\Rightarrow S_1(t) = \frac{5}{6} + \sum_{n=1}^{\infty}\frac{-5}{3\pi n}\sin\frac{2\pi n}{5}t$

A1.3 - Changing of trigonometric and exponential coefficients

The function S2(t) is the same as S1(t) but it has been shift upwards by 2. Therefore, the shape of the function has not been interfered with. Therefore the trigonometric and exponential coefficient that approximate this shape will be the same. However, the coefficient a0 nad c0 which describe the DC offset are increased by 2 as the original function has been shift up by 2.

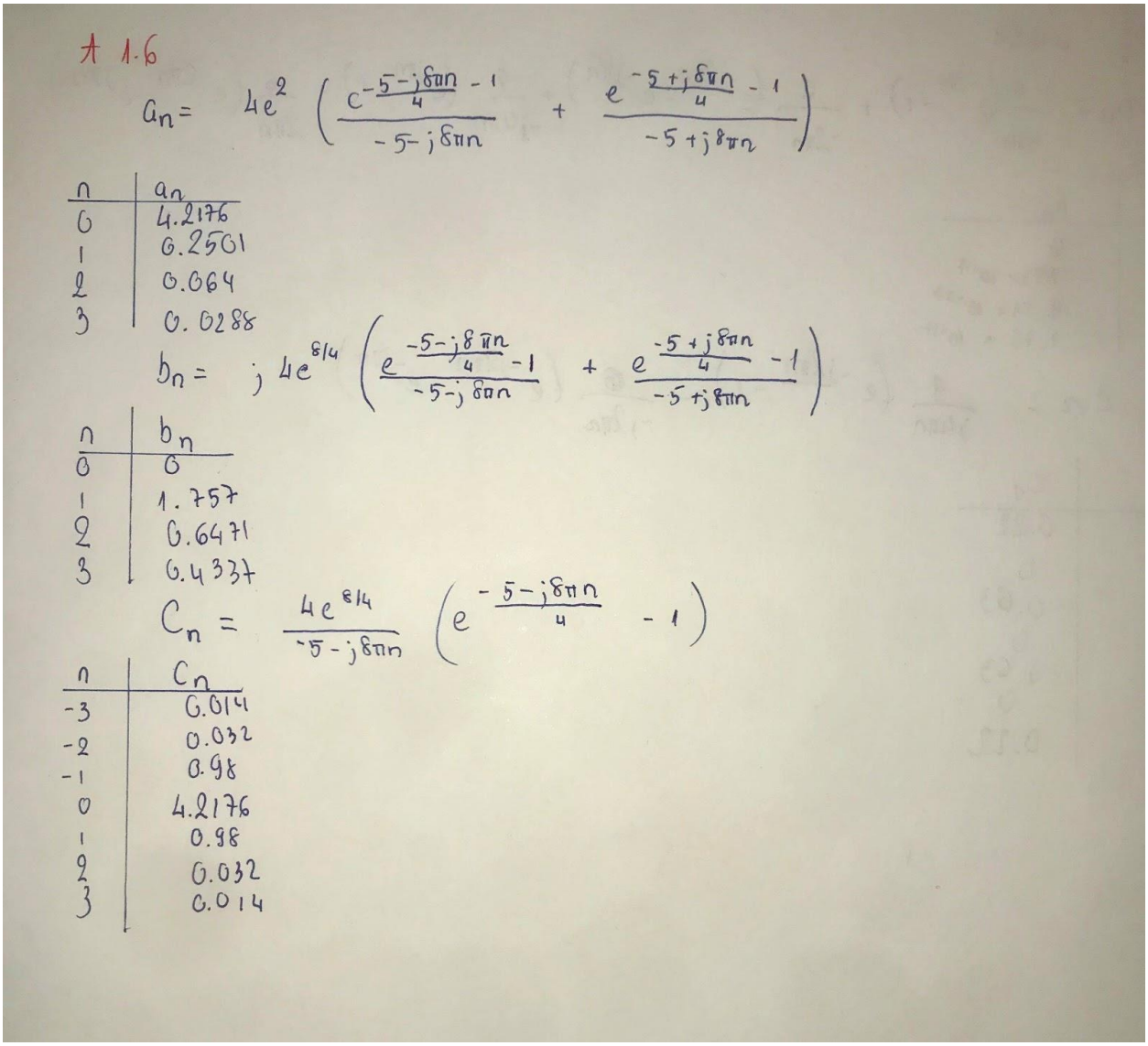
A1.4 - Signal expansion



A1.5 - Choice of Fourier series for first principle expansion of s2(t)

The complex fourier series was chosen for the first principal expansion of S2(t). The complex exponential Fourier series was chosen as S2(t). The complex exponential Fourier series was chosen as S2(t) is an exponential function. The exponential of the function and the exponential of the function and the exponential of the equation use to calculate the coefficient can be simplified to make the calculation easier. In addition, complex Fourier series coefficient are faster to calculate by hand than trigonometric Fourier series coefficients. This is because only one integration need to be completed to attain Cn whereas two integration need to be completed in trigonometric fourier series to calculate An and Bn.

A1.6 - TFS and EFS coefficients for s2(t)



A 1.6

$$a_n = 4e^2 \left(\frac{e^{\frac{-5-j8\pi n}{4}} - 1}{-5-j8\pi n} + \frac{e^{\frac{-5+j8\pi n}{4}} - 1}{-5+j8\pi n} \right)$$

n	a_n
0	4.2176
1	0.2501
2	0.064
3	0.0288

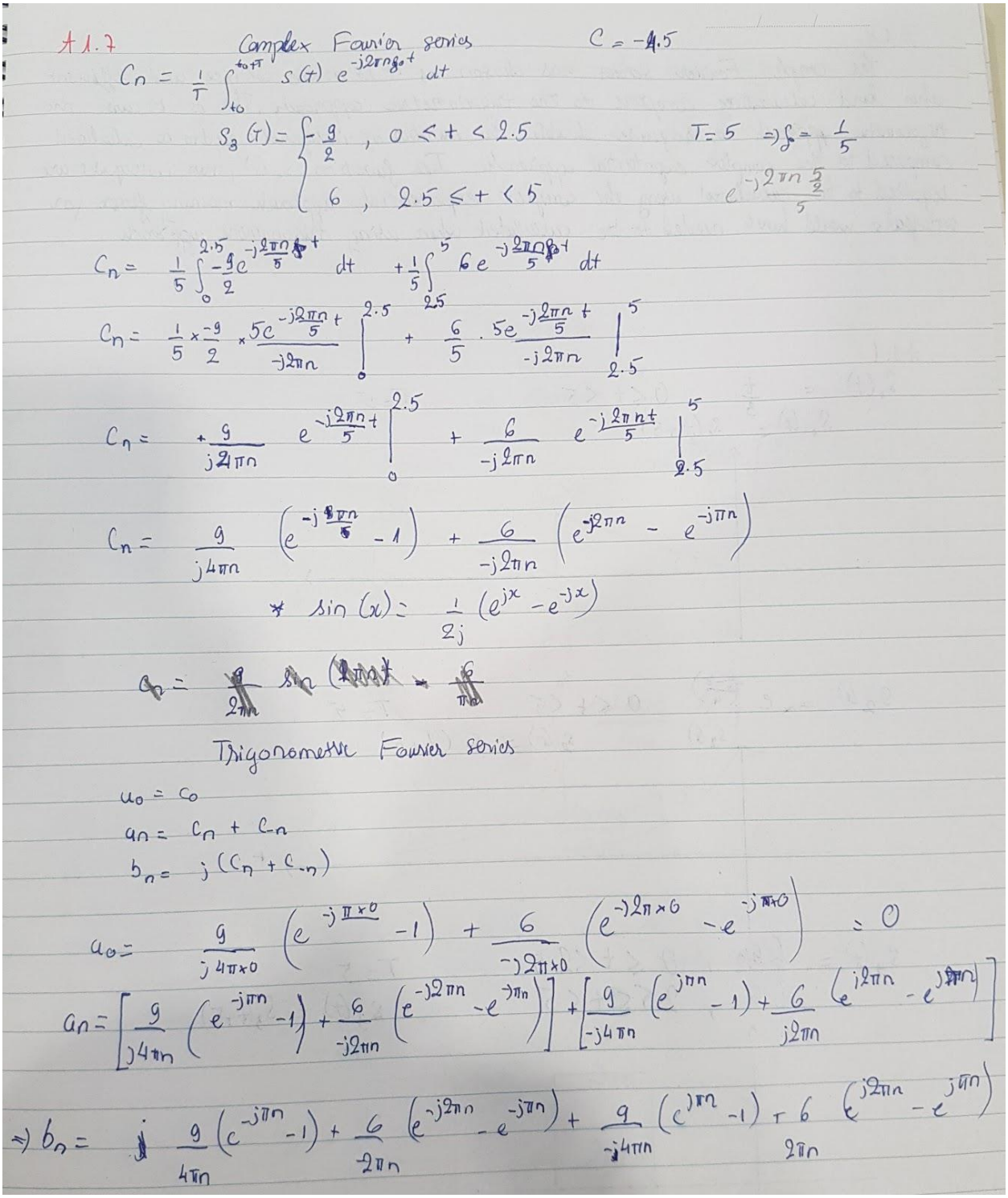
$$b_n = j4e^{8/4} \left(\frac{e^{\frac{-5-j8\pi n}{4}} - 1}{-5-j8\pi n} + \frac{e^{\frac{-5+j8\pi n}{4}} - 1}{-5+j8\pi n} \right)$$

n	b_n
0	0
1	1.757
2	0.6471
3	0.4337

$$C_n = \frac{4e^{8/4}}{-5-j8\pi n} \left(e^{\frac{-5-j8\pi n}{4}} - 1 \right)$$

n	C_n
-3	0.014
-2	0.032
-1	0.98
0	4.2176
1	0.98
2	0.032
3	0.014

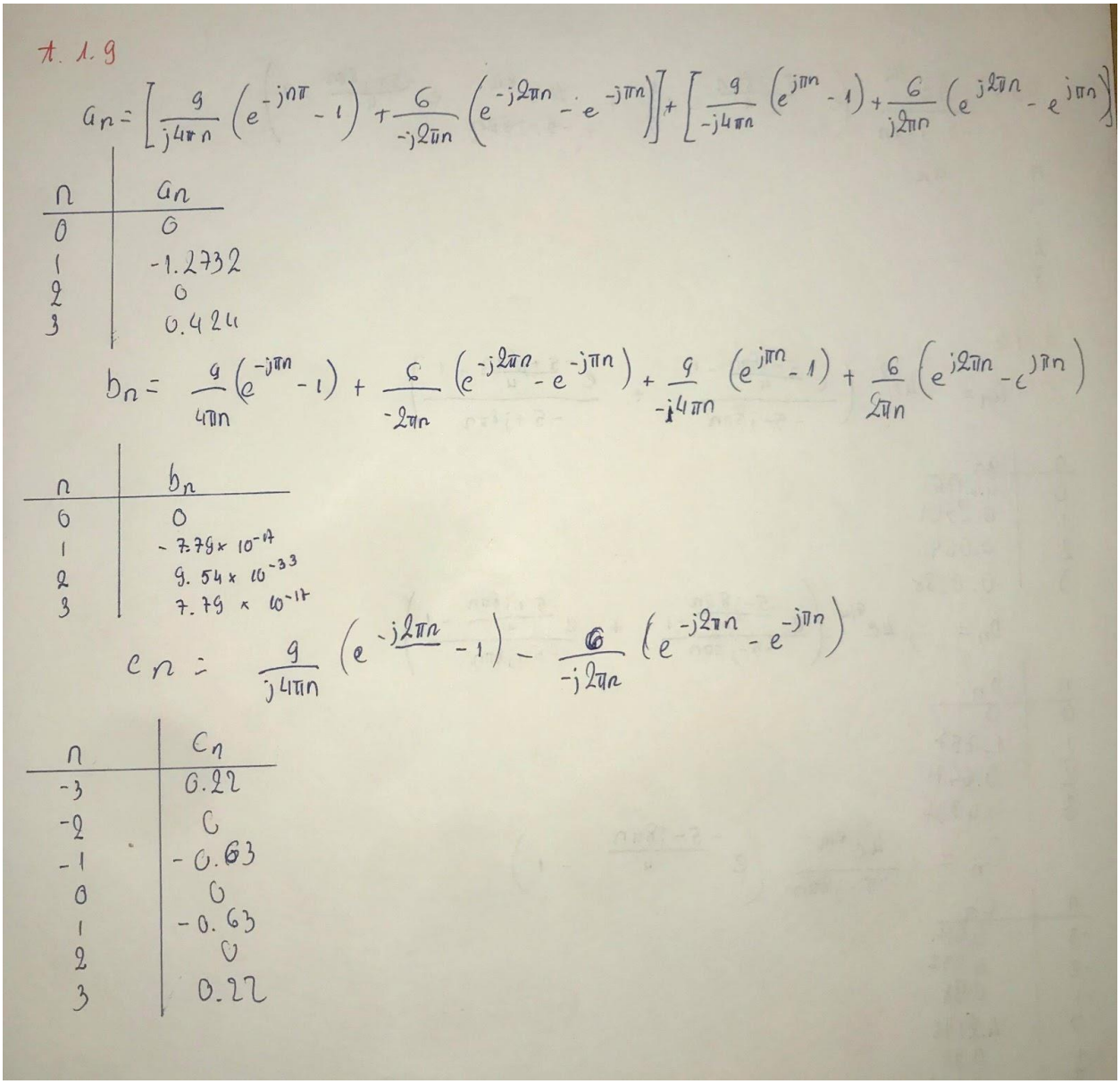
A1.7 - Fourier series of s3(t)



A1.8 - Choice of Fourier series for first principle expansion of s3(t)

The complex Fourier series was chosen as it is more concise and efficient when hand calculating compare to the trigonometric approach. This is because the trigonometric approach to requires double the number of integral need to be calculated compared to the complex exponential approach. In function S3(t), two integrals are required to be calculated using the complex exponential approach meaning for integral would have need to be calculated when using trigonometric.

A1.9 - TFS and EFS coefficients for s3(t)



A 1.9

$$a_n = \left[\frac{9}{j4\pi n}\left(e^{-jn\pi} - 1\right) + \frac{6}{-j2\pi n}\left(e^{-j2\pi n} - e^{-j\pi n}\right)\right] + \left[\frac{9}{-j4\pi n}\left(e^{j\pi n} - 1\right) + \frac{6}{j2\pi n}\left(e^{j2\pi n} - e^{j\pi n}\right)\right]$$

n	a_n
0	0
1	-1.2732
2	0
3	0.424

$$b_n = \frac{9}{4\pi n}\left(e^{-j\pi n} - 1\right) + \frac{6}{-2\pi n}\left(e^{-j2\pi n} - e^{-j\pi n}\right) + \frac{9}{-j4\pi n}\left(e^{j\pi n} - 1\right) + \frac{6}{2\pi n}\left(e^{j2\pi n} - e^{j\pi n}\right)$$

n	b_n
0	0
1	-7.79×10^{-17}
2	9.54×10^{-33}
3	7.79×10^{-17}

$$c_n = \frac{9}{j4\pi n}\left(e^{-j2\pi n} - 1\right) - \frac{6}{-j2\pi n}\left(e^{-j2\pi n} - e^{-j\pi n}\right)$$

n	c_n
-3	0.22
-2	0
-1	-0.63
0	0
1	-0.63
2	0
3	0.22

A1.10 - Classifying test signals

A 1.10

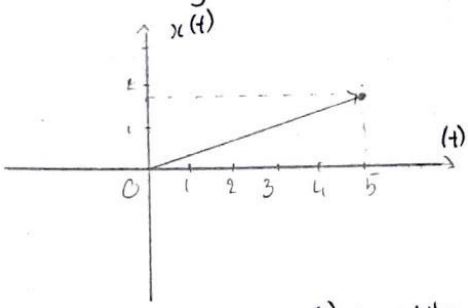
Even signal : $x(t) = x(-t)$

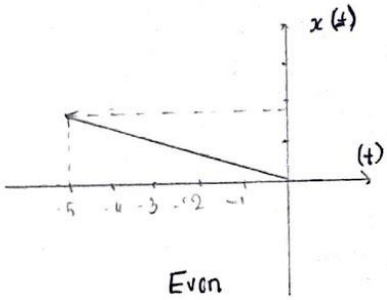
Odd signal : $x(t) = -x(t)$

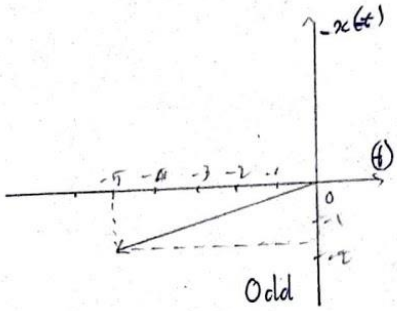
Neither even nor odd signal

$[x(t) \neq x(-t)$
$x(t) \neq -x(-t)$

* $S_1(t) = \frac{t}{3}$ $0 \leq t < 5$

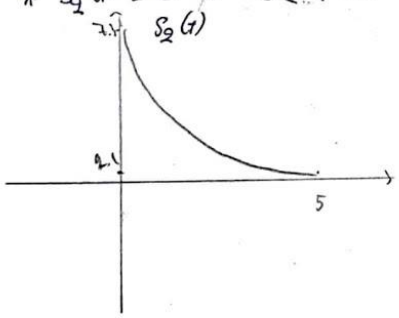


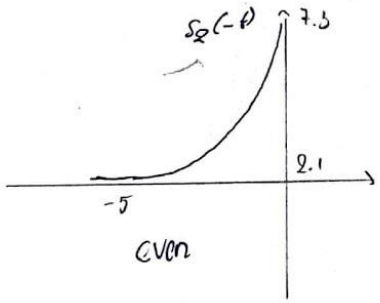


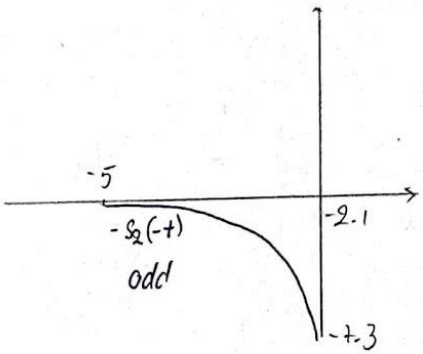


$\Rightarrow S_1(t)$ is neither even nor odd

* $S_2(t) = e^{\left(\frac{-t-8}{4}\right)}$ $0 \leq t < 5$

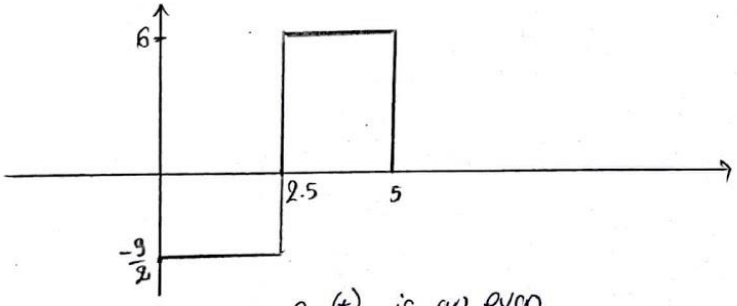






$\Rightarrow S_2(t)$ is neither even nor odd

* $S_3(t) = \begin{cases} -\frac{9}{2} & 0 \leq t < 2.5 \\ 6 & 2.5 \leq t < 5 \end{cases}$



$\Rightarrow S_3(t)$ is an even

A2 - BASA Headquarters: Training Exercise

This section deals much of the periodic signals on section A1, and aims to produce the signals live on MATLAB, generating plots of test signals and seeing them approximate through different levels of harmonics. The methods of these approximations are dealt with trigonometric and complex exponential fourier series.

Each periodic function defined in the test signal definition from A1 are a single period and are coded to have 100 samples. The period (T) for all functions is to be 5 seconds, and frequency (f0) of 1/T is made to calculate the trigonometric and exponential coefficients. A single period time vector (x) is established to plot a period of the signal. A time vector (t) is made to support plotting periodic signals which is 5 times bigger than the period time vector, also being able to hold 500 samples. Finally, a variable to store a number of harmonics is created to show different signal approximations and a sampling frequency is set up to support calculations of the coefficients needed.

```
33      %% A2.0 - Setting up common variables for preparation.m
34 -    T = 5; %period (t)
35 -    f0 = 1/T; %frequency
36 -    samples = 1e2; %100 points per period
37
38 -    x = linspace(0,T,samples + 1); x(end) = []; %1 period time vector
39 -    t = linspace(0,5*T,5*samples + 1); t(end) = []; %time vector
40
41 -    numHarm = 3; %no. of harmonics for Fourier series approximation
42 -    fs = 1/(x(2) - x(1)); %sampling frequency
43
44 -    if numHarm <= 0
45 -       error('Number of Harmonics must be greater than 0.');
46 -    end
```

A2.1 - Creating a periodic signal based on s2(t)

A variable is made for the signal s2(t), substituting the single period time vector and one of the generated values (where B = 8) in the periodic function.

```
49 -    s2 = exp(-(x-B)/4); %s2(t)
```

Variable ‘s2_hinf’ is the ideal time series representation of the signal that allows to create 5 periods of the function.

```
54      %signal period spanning 5 cycles
55 -    s2_hinf = repmat(s2,[1 5]);
```

By further expanding this signal for 5 cycles (Figure 2.1), each sample will therefore contribute to a total of 500 sample points, and a time vector will therefore be used to hold the amount of sample points needed to generate a longer signal.

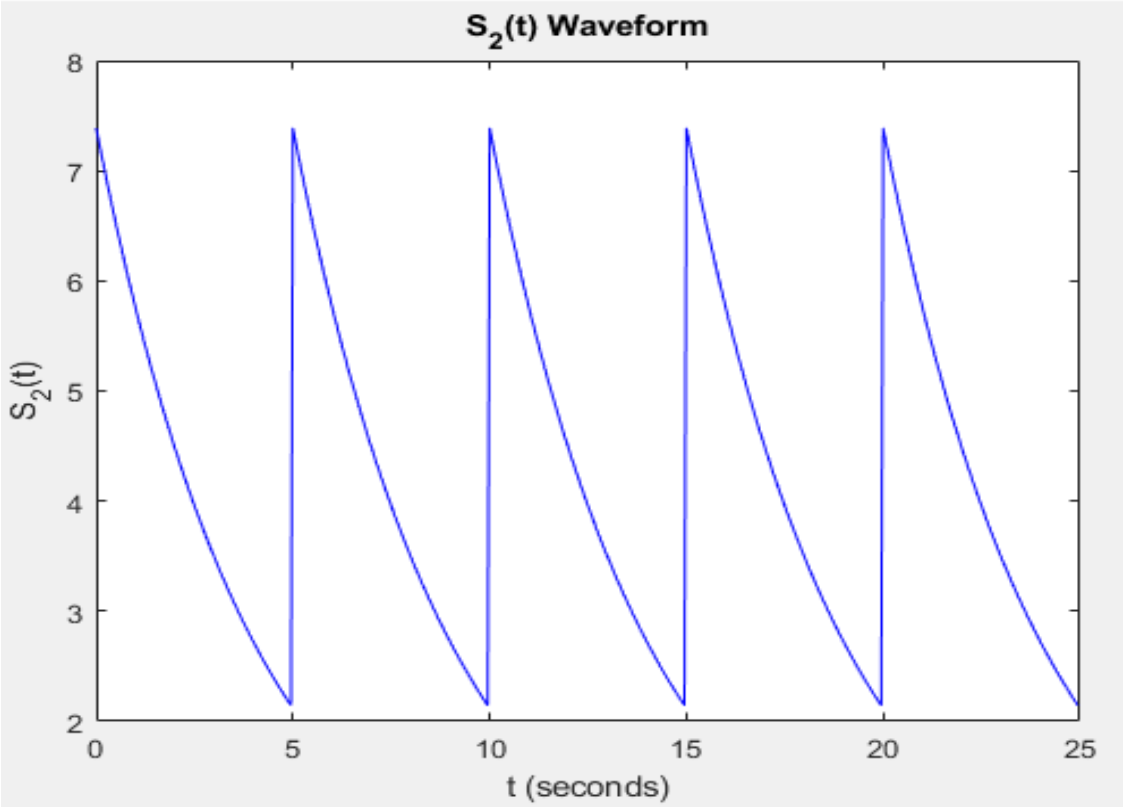


Figure 2.1: The periodic signal of s2(t).

A2.2 - Trigonometric coefficients of s2(t) using MATLAB

It is possible to compute the signal function of s2(t)'s trigonometric coefficients without using the 'trapz' or 'syms' functions. The process of using TFS involves setting up a number of harmonics which for this case is 3, and implementing the mathematical equations of a0, an, and bn. The coefficient values will only calculate a single period (i.e. 100 sample points).

```
65 -    n_trig = (1:numHarm).'; %no. of harmonics for TFS
66 -    wnt = 2 * pi * f0 * (n_trig * x);
67
68      %trigonometric coefficients for s2(t)
69 -    a0 = (1/T) * sum(s2) / fs;
70 -    an = (2/T) * s2 * cos(wnt).' / fs;
71 -    bn = (2/T) * s2 * sin(wnt).' / fs;
```

A2.3 - A 4x500 matrix to represent a0 and three different harmonics

In order to obtain different series of harmonics, a matrix with 4 rows and 500 columns is created to hold up to three harmonics, and also the dc offset values as well. The first row contains a constant value of 'a0', while the three remaining rows will hold a single harmonic component of the signal, where n is the element of 1,2,3 and a0's n is the element of 0.

While the matrix is established, a for loop that iterates 3 times is used to calculate the trigonometric fourier series of each n element (for harmonic component = 1,2,3). Each series that is calculated then gets its vector repeated 5 times to match the size dimensions of the matrix, expanding it to 500 columns of values. This value of rows is now inserted in the matrix, and iterates to a new series.

```
74 -  for ii = 1:length(n_trig)
75 -      xtf = an(ii)*cos(2*pi*f0*n_trig(ii)*x) + bn(ii)*sin(2*pi*f0*n_trig(ii)*x); %single harmonic component
76 -      h_component = repmat(xtf,[1 5]); %expanding vector to 500 samples
77 -      s2_matrix(ii,:) = h_component; %inserting harmonics to each row for n = 1,2,3
78 -  end
79
80 -  row1(1:1,1:500) = a0; %DC component for first row (n = 0)
81 -  s2_matrix = [row1; s2_matrix]; %final 4x500 matrix
```

A2.4 - Approximation of signal s2(t)

A vector to hold the approximation of the signal s2(t) is created by the sum of each row in the s2_matrix. This approximation is based on 3 harmonics.

```
84 -    s2_approx = sum(s2_matrix);
```

A2.5 - Plotting Trigonometric Fourier series approximation of s2(t)

Representations of different signal approximations can be derived from different harmonic levels from the s2_matrix. Using for loop, a0 is used to add the fundamental frequency, which concludes the first signal approximation. For the second approximation, it is a0 + fundamental frequency + second harmonic, and finally the third approximation which combines all harmonic components and the dc component. A matrix collects each approximation, totalling up to 3 per row and removing the first empty row due to number iteration differences with s2_matrix.

```
91      %computing signal approximations
92 -    n = a0; %DC component
93 -  for ii = 2:length(n_trig)+1 %iterating from 2nd row of matrix -> last row
94 -      n = n + s2_matrix(ii,:); %signal approximation
95 -      s2_n(ii,:) = n; %no. of approximations of s2(t)
96 -  end
97 -  s2_n(1,:)=[]; %remove empty row
```

In addition to this, a plot (Figure 2.2) is made to round up all the signal approximations in one graph, alongside with the ideal periodic signal. Each signal is colour coded based on the legend and represents 3 different harmonic signals.

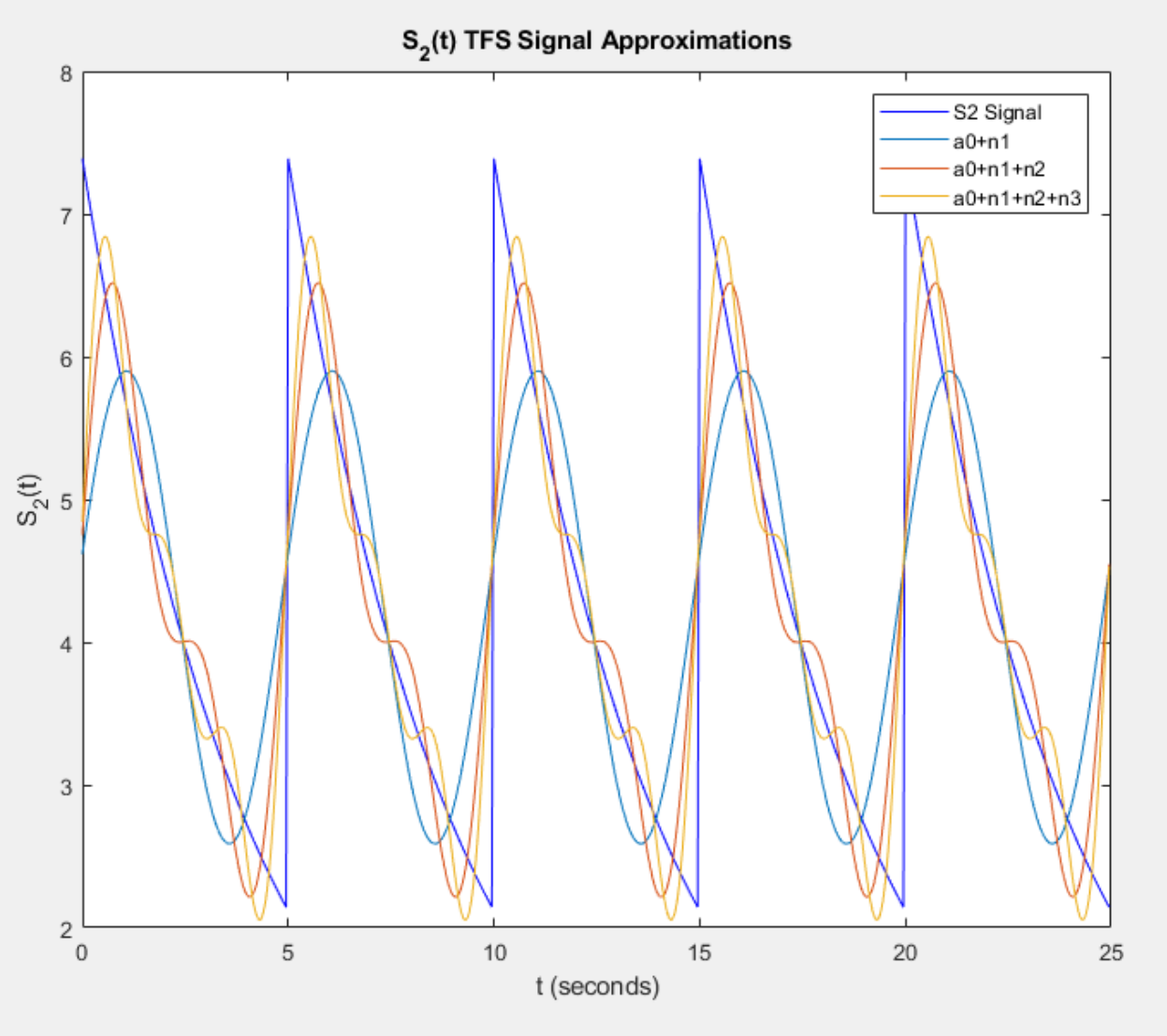


Figure 2.2: The Fourier series approximation using trigonometric coefficients.

A2.6 - Modelling test signal s3(t)

A new signal is generated and plotted from s3(t), which also has a period of 5 seconds and the same frequency and sampling per period. The periodic function has 2y points: where C = -4.5 being the minimum y point that travels from 0 to 2.5 seconds, and a maximum y point of 6, travelling from 2.5 to 5 seconds.

```
107        %s3(t): consider 2 yl points
108 -      s3 = zeros(size(x));
109 -      s3(x>= 0 & x<2.5) = -4.5;
110 -      s3(x>=2.5 & x<5) = 6;
```

Furthermore, the signal is to span 5 cycles of its original form by using the recognised 'repmat' function. This allows the following plot (Figure 2.3) to be made, showing the periodic function at its ideal time series representation.

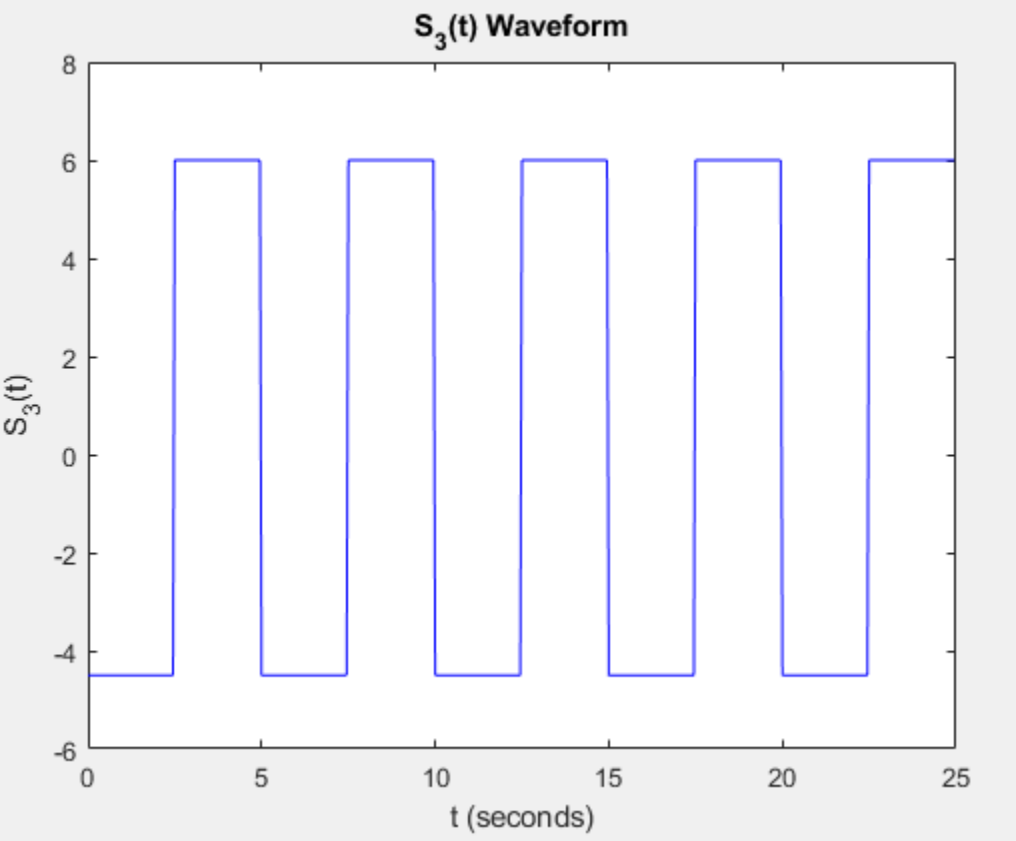


Figure 2.3: The periodic signal of s3(t).

Another method to calculate Fourier series is using complex exponential coefficients. In addition, the recently used trigonometric coefficients are used to be part of the complex variables, where the 'cn' of positive and negative acquire both 'an' and 'bn' variables and where 'c0' is already established to be 'a0'.

```
131        %trigonometric coefficients for s3(t)
132 -      a0 = (1/T) * sum(s3) / fs;
133 -      an = (2/T) * s3 * cos(wnt).' / fs;
134 -      bn = (2/T) * s3 * sin(wnt).' / fs;
135
136        %exponential coefficients for s3(t)
137 -      c0 = a0;
138 -      cn_pos = 1/2 * (an - 1i * bn);
139 -      cn_neg = 1/2 * (an + 1i * bn);
```

```
141 -    n_comp = (-numHarm:numHarm).';
142      %[c-3 c-2 c-1 c0 c1 c2 c3];
143
144 -    cn = [fliplr(cn_neg), c0, cn_pos]; %range of cn values
145
146 -    table(n_comp,cn.','VariableNames',{'n','Cn'}); %table view
```

Unlike n_trig where number of harmonics is dealt from 1:3, the exponential version of 'n_comp' uses harmonics to range from -3:3. This is where the negative cn values are stored on the negative range, where c0 is in the middle, and the positive cn values go to the positive range. A table (Figure 2.4) is created to illustrate the n and cn columns, listing all cn values and their respective n range.

n	Cn
-3	-0.105-1.1108i
-2	-3.2203e-16-2.5397e-16i
-1	-0.105-3.3412i
0	0.75+0i
1	-0.105+3.3412i
2	-3.2203e-16+2.5397e-16i
3	-0.105+1.1108i

Figure 2.4: A table listing n_comp and cn values.

A 7x500 matrix is created to store all the cn values of the table on each row. Alongside with this, a for loop iterating from 1 to the length of n_comp which is 7, calculates the complex fourier series of each row of the matrix, collecting 500 sample values for each of them.

```
148      %creating a 7x500 matrix called s3_matrix
149 -    for ii = 1:length(n_comp)
150          %complex fourier series
151 -        s3_matrix(ii,:) = cn(ii) * exp(1j * 2 * pi * f0 * n_comp(ii) * t);
152 -    end
```

A vector called s3_approx for signal approximation is created to collect the sum of all values of each column, representing the third approximation of the signal.

```
154      %signal approximation values for s3
155 -    s3_approx = sum(s3_matrix);
```

A fourier series approximation plot of different harmonics is created to display the development of the signals using exponential coefficients. These include:

- An approximation of s3(t) using the DC component and the fundamental frequency,
- An approximation of s3(t) using the DC component, the fundamental frequency and the second harmonic, and
- An approximation of s3(t) using the DC component, the fundamental frequency and the second and third harmonics.

```
157      %computing signal approximations
158 -    for ii = 1:length(n_comp)/2
159 -        harmonics(ii,:) = real(s3_matrix(ii,:) * 2); %single harmonic component
160 -    end
161
162 -    n = c0; %DC component
163 -    for ii = ((length(n_comp) -1)/2):-1:1 %iterating no. of harmonics -> 1
164 -        n = n + harmonics(ii,:); %signal approximation
165 -        s3_n(ii,:) = n; %no. of approximations of s3(t)
166 -    end
167 -    s3_n = flipud(s3_n); %fix legend
```

The approximations are made by associating harmonics of the same n value of positive cn and negative cn. By summing these together, they can now become a single harmonic component, ready to be added to c0, where the fundamental frequency (harmonic one) for example, is elements of n = 3 and 5 of s3_matrix. The second signal approximation is done by having c0 + single harmonic component 1 and 2, and the third signal approximation is a result of c0 + all three harmonic components. A plot is created to display (Figure 2.5) all of the approximations of s3(t).

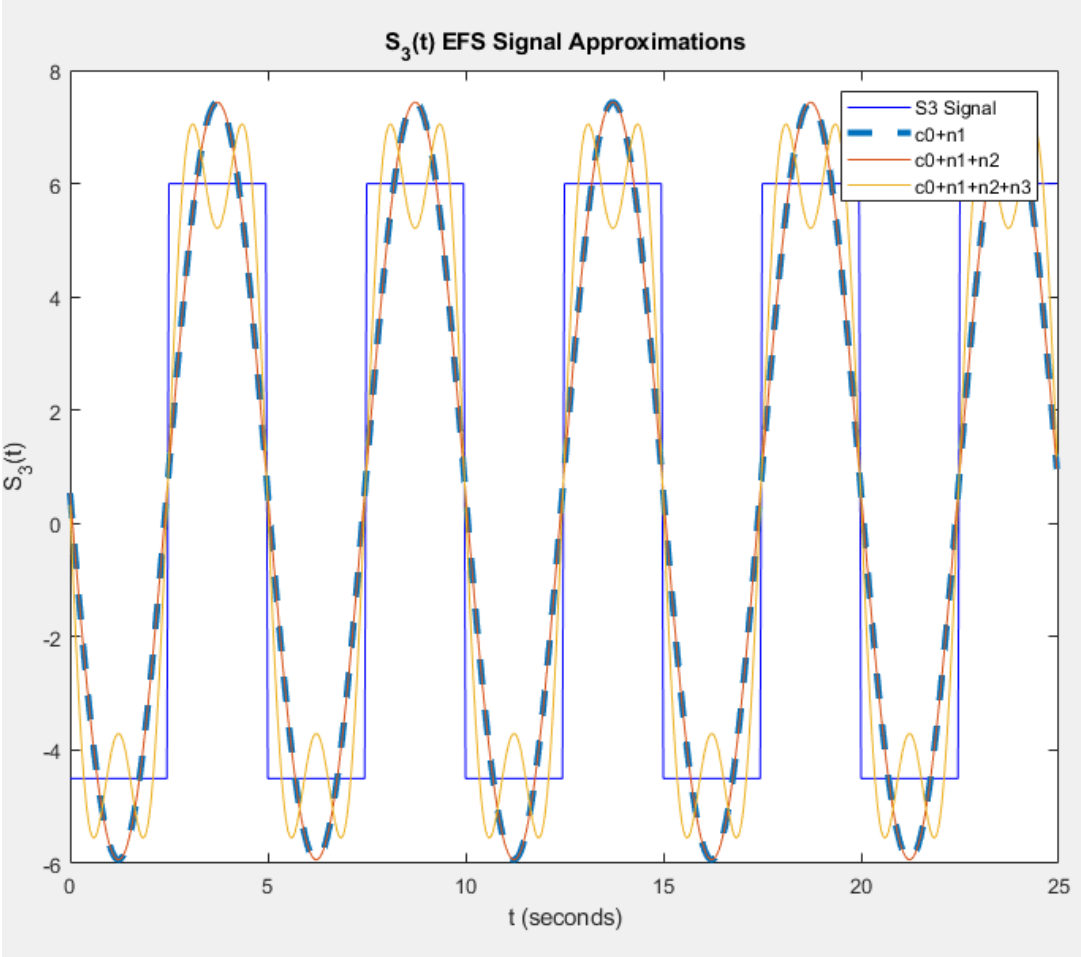


Figure 2.5: *The Fourier series approximation using exponential coefficients.*

A2.7 - Approximation Results

The number of harmonics applied increases the approximation accuracy of the signal. The current number of harmonics (3) is sufficient enough to demonstrate the signals and its approximation, and lowering the number of harmonics would not prove so much or display an accurate measurement of the signal, creating a broad interpretation of the information as shown in s3(t) signal approximation graph (Figure 2.5) where the first two signal approximations are the same. Raising the number of harmonics increases accuracy and show many transformations, however it would further narrow down the waveforms of the signal, making it smaller until it turns to the ideal periodic series.

A3 - Mars Mission: De-noising Speech

BASA has received a corrupted message sent from the team of MARS-242 astronauts heading to Mars. The following section analyses the speech signal that has been corrupted by an additive noise process, and is in need to be solved with the aid of MATLAB.

A3.1 - Noise waveform identification

As part of this assignment we have been provided with three test signals:

$$S_1(t) = \frac{t}{A}, 0 \leq t < 5$$

$$S_2(t) = e^{-(\frac{t-B}{4})}, 0 \leq t < 5$$

$$S_3(t) = \begin{cases} C, 0 \leq t < 2.5 \\ 6, 2.5 \leq t < 5 \end{cases}$$

As we can see from section A1.1, $S_3$is a square wave signal. This corresponds with a test plot we did of our noiseSound variable in the figure below.

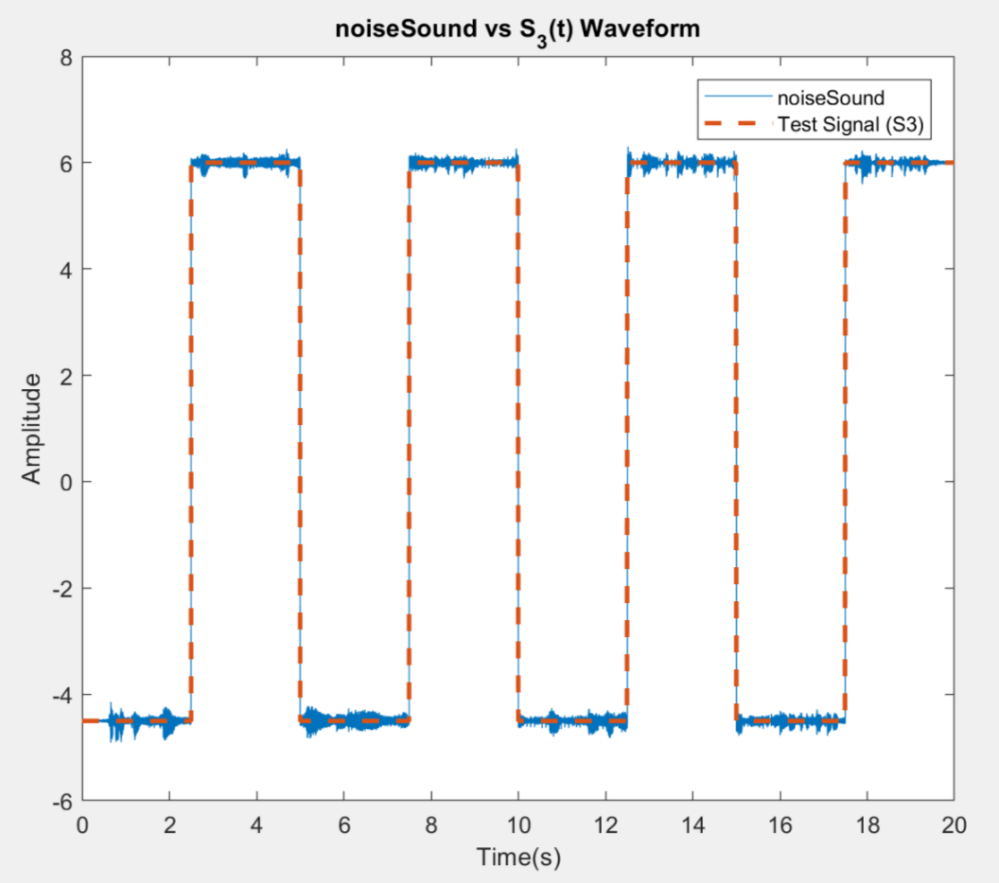


Figure 3.1: The noiseSound signal vs S3 periodic signal.

As we can see these two waveforms match extremely well compared to S_1 and S_2 with a period of 4 and a similar amplitude the team decided to use S_3 as the ideal waveform. Below you can see a close up sample of the signal received by the team.

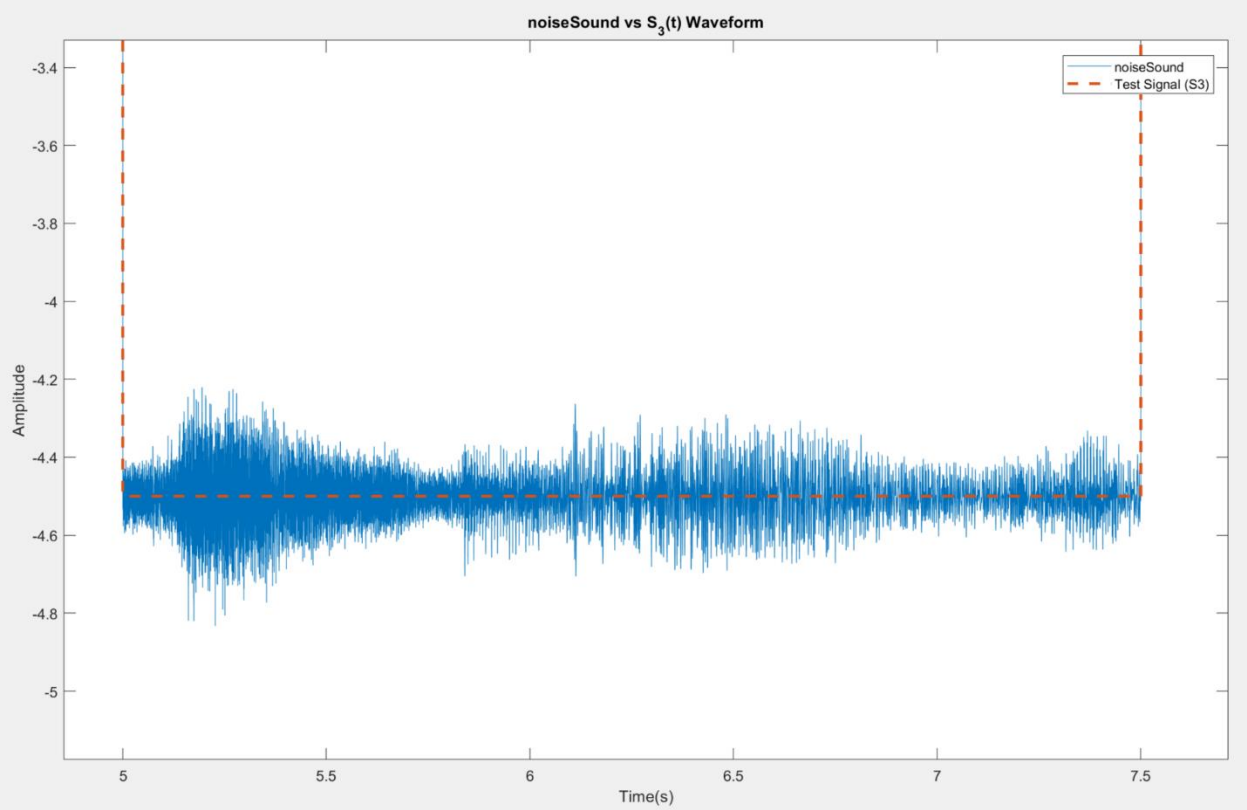


Figure 3.2: The periodic noise of the corrupted speech.

A3.2 Noise waveform generation

In the below snip of code we can see the setting of the variables used to generate the noise waveform.

```
30 -    T = 5; %Period
31 -    f0 = 1/T; %frequency
32 -    samples = 44100; %rate per second
33
34 -    x = linspace(0, T, 5*samples + 1); x(end) = []; %Time vector (5s)
35 -    t = linspace(0, 4*T, 20*samples + 1); t(end) = []; % t - Time vector (20s)
36
37 -    Ts = t(2) - t(1); %sampling frequency
```

Hence, we save noiseSound to a new variable called ‘additive_noise’.

```
62 -    additive_noise = noiseSound;
```

A3.3 - Evaluating coefficients of noise signal using MATLAB

In the below snip of code we can see the code used to generate our Complex Fourier series for c0 and cn for $-10 \leq n \leq 10$:

```
numHarm = 10;
%setting up trigonometric coefficients for ESF
n_trig = (1:numHarm).';
a0 = (1/T)*sum(s3)*Ts;
an = (2/T)*s3*cos(2*pi*f0*n_trig*x).'*Ts;
bn = (2/T)*s3*sin(2*pi*f0*n_trig*x).'*Ts;

%apply variables to complex coefficients
c0 = a0;
cn_pos = 1/2*(an - 1i*bn);
cn_neg = 1/2*(an + 1i*bn);
n_comp = (-numHarm:numHarm).';
cn = [fliplr(cn_neg), c0, cn_pos];
```

'S3'is defined in section A3.1 where it was used to identify the noise signal. The function is used to help compute the Fourier series.

```
%set up s3
s3(x<=2.5) = C;
s3(x>2.5) = 6;
s3_hinf = repmat(s3,[1 4]); %cycles 4 times
```

A3.4 - Code to generate Fourier series approximation

A for loop is used to calculate the Complex series iterated by the number of harmonics. A matrix is used to collect all the series and is then summed up to get the signal approximation of the noise signal.

```
for ii = 1:length(n_comp)
    fs1_app(ii,:) = cn(ii) * exp(1j * 2 * pi * f0 * n_comp(ii) * t);
end

FS1 = sum(fs1_app); %signal noise approximation
```

As a result, our plot (Figure 3.3) shows what the approximated signal of the noise would look like, soon to be used to minimalise the noise with the number of harmonics.

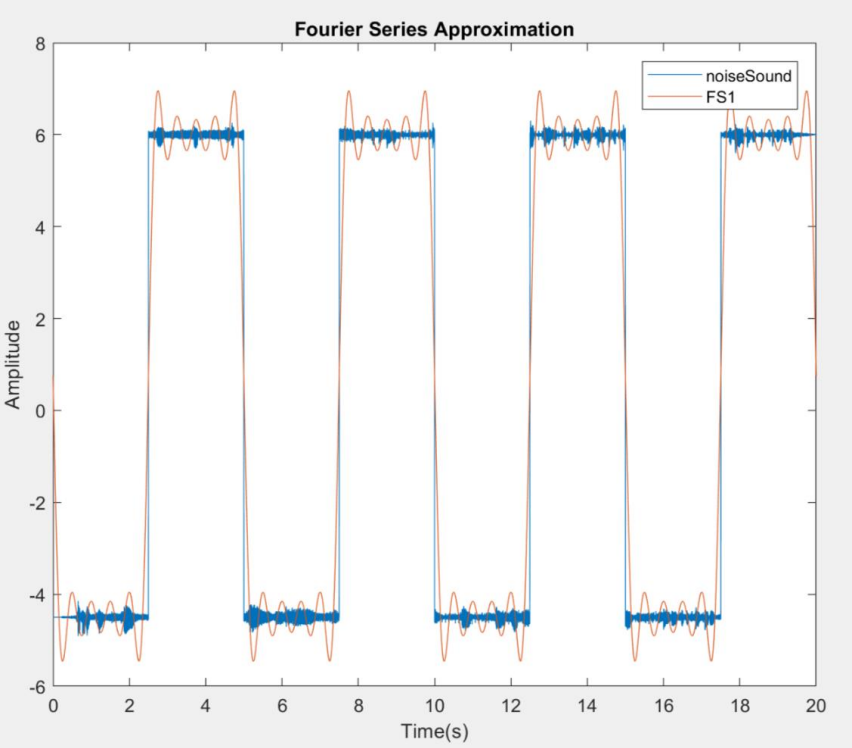


Figure 3.3: Complex Fourier series signal approximation.

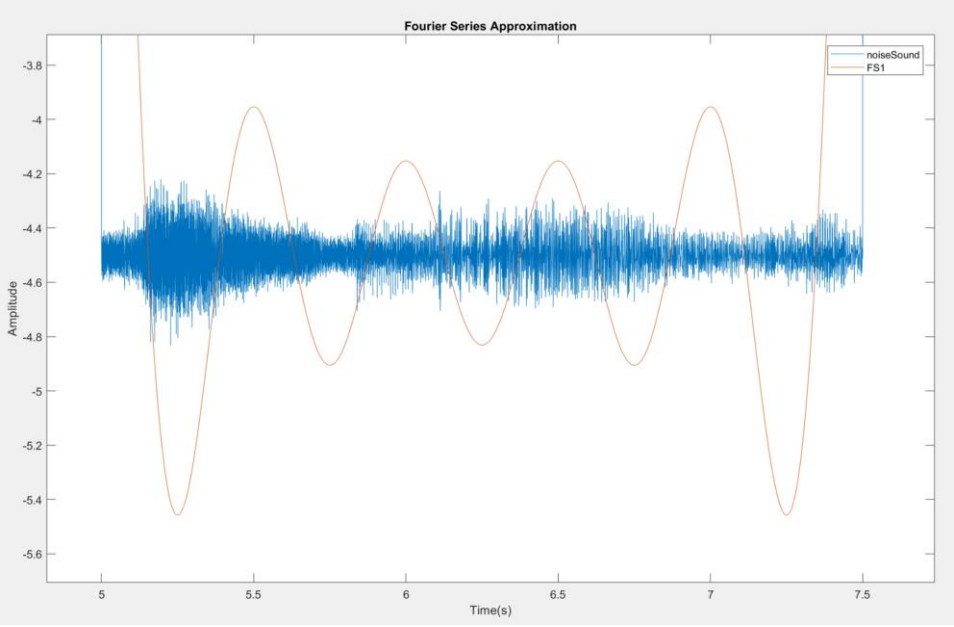


Figure 3.4: Close up of the signal noise and complex series.

A3.5 - Recovering corrupted speech

Recovering the corrupted speech proved to be the easiest part and was a simple subtraction formula as seen from the code snip below.

```
94      %Transpose to minimise matlab resources
95 -    additive_noise = additive_noise';
96      %de-noised result
97 -    dnSnd = additive_noise-FS1;
```

A3.6 - Display of recovered speech

We can see below that the de-noised waveform contains sharp peaks in a more common period than our previous waveforms, we can still see these peaks are reminiscent of our square waveform however. These peaks could attribute for the “pops” we hear in playing the resulting sound file.

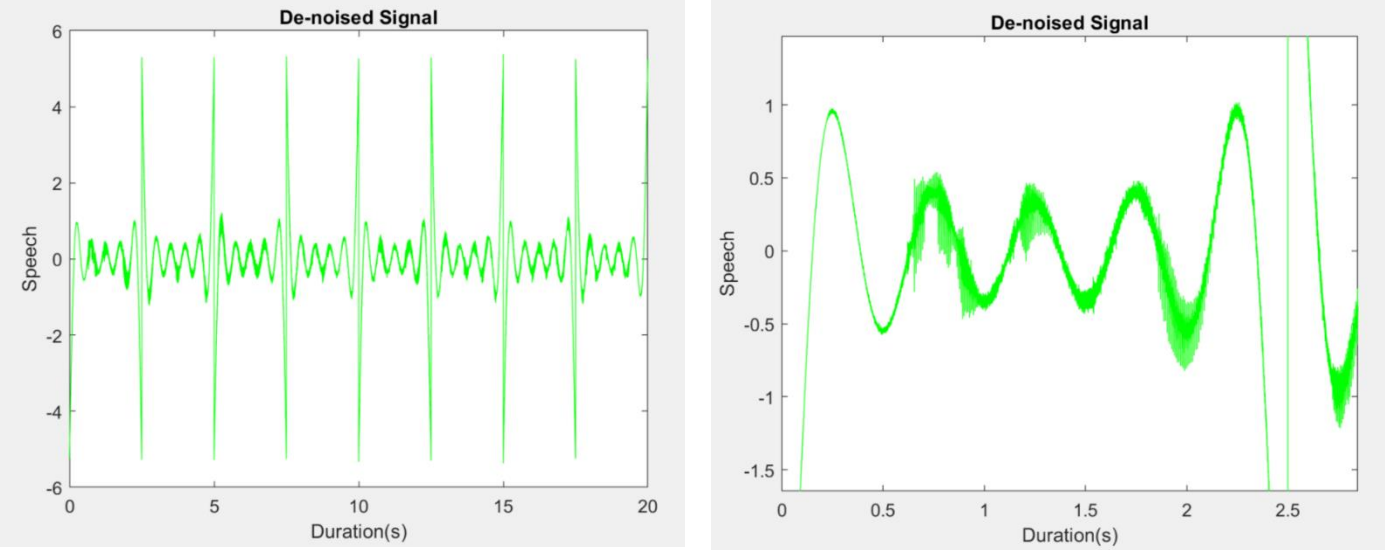


Figure 3.5: Recovered speech result.

As we can see on examining the close up on the right, the Fourier approximation and de-noising the signal has essentially transformed our data input into a digital output. This has been done by using a Fourier approximation to represent the periodic signal by sum of sine and cosine terms, then removing the interfering signal and reconstructing the signal. We can see that the final signal is much clearer and louder than the initial.

Below is a transcript of the received signal:

Transcript:

T-minus 10, 9, 8, 7, 6, 5. All three engines up and burning! 2, 1, 0 and lift off! The final lift off of Atlantis! On the shoulders of the space shuttle, America will continue the dream.

A4 - Reflection

The speech signal was recovered from the noisy speech signal by subtracting the additive noise from the noisy speech signal. To do this we approximated the noisy speech signal with a Fourier series approximation using trigonometric coefficients. This provided us with a function that described the changes in amplitude over the correct frequencies as the noisy speech signal. After subtracting this approximation from the noisy speech signal, we could remove most of the unwanted changes in amplitude that were occurring over the identified frequency. Although, the de-noised signal still contained large jumps in amplitude over small time intervals that would occur along the same frequency as the noise. To reduce this effect, more harmonics need to be used when approximation the noisy speech signal to remove even more unwanted amplitudes across that frequency. Therefore, Fourier series approximations can be practically used to approximate a signal that has unwanted amplitude that occurs over a constant frequency. This approximation can then be subtracted from the original signal to reduce or remove the unwanted amplitudes across the identified frequencies.

Bryan: in completing this assignment, we have learnt lessons about how to produce Fourier series approximations; when to use a trigonometric or complex exponential approach and what Fourier series approximations can be used for. We have also gained a range of skills within MATLAB based around the calculations of Fourier series approximations within the program. If we were to attempt this again we would start with a greater understanding of the topic; have more detail in workload planning; and communicate better as a group. **Kevin:** I've learnt a lot today about the uses of trigonometric and complex series and how they both use different methods to get the same results. It showed me how different levels of signal approximations greatly reflect to the ideal signal, getting more and more accurate when using lots of harmonics. To further improve my code, I would've made the legend dynamic to include more different level of harmonics for each signal approximation, using for loop to display the information and use more LineStyles, and thickness to distinguish many signals. **Callum:** This report has given me an appreciation for the math behind the electronics and signals we use in day to day life. If I had to attempt this again and had a greater knowledge of matlab, I would attempt to perhaps use an if function or some unknown matlab feature to remove the outliers that caused the high peaks that caused the loud pops heard in the sample for section 3.