Report

SQUARE WAVE GENERATOR

Written by: Nguyen Ngoc Minh – e1601125

Submit date: 30/03/2018

Table of Contents

[1. Abstract 3](#_Toc510216543)

[2. Introduction 3](#_Toc510216544)

[3. Procedure 3](#_Toc510216545)

[3.1. Defining inputs 3](#_Toc510216546)

[3.2. Putting inputs into the equations 4](#_Toc510216547)

[Calculate 4](#_Toc510216548)

[Calculate 4](#_Toc510216549)

[Calculate 5](#_Toc510216550)

[3.3. Function definition 5](#_Toc510216551)

[Function 1: square\_wave\_project 5](#_Toc510216552)

[Function 2: get\_default\_options 6](#_Toc510216553)

[Function 3: make\_square\_wave 6](#_Toc510216554)

[Function 4: visualization 7](#_Toc510216555)

[4. Result 7](#_Toc510216556)

[5. References 10](#_Toc510216557)

# Abstract

In this project, I attempt to use Fourier series expansion to draw a square wave from a sum of harmonics in Matlab.

# Introduction

Formulas:

Fourier series expansion:

(1)

Calculate (“start” is a starting value, T is the period of the function):

(2)

Calculate :

(3)

Calculate :

(4)

# Procedure

## Defining inputs

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Description | Value range | Default value |
| T | Period of the function (sec) |  | 1 |
| num | Maximum of n |  | 100 |
| max | High value of the function |  | 5 |
| min | Low value of the function |  | -5 |
| start | Starting time (sec) |  | 0 |
| state | Starting state | 0 or 1 | 0 |

Figure 1. Inputs table

## Putting inputs into the equations

### Calculate

Since this is a square wave, in one period, half of the time , and the other half . The value of and depend on “state”, so:

Below is the truth table of the above two equations:

|  |  |  |
| --- | --- | --- |
| state |  |  |
| 1 | max | min |
| 0 | min | max |

Figure 2. Truth table of variable “state”

The equation becomes:

### Calculate

Since ,

### Calculate

Since ,

## Function definition

### Function 1: square\_wave\_project

Usage: square\_wave\_project ([T, [num, [max, [min, [start, [state]]]]]])

Description: This function can be thought of as a main function. It should be called from the Command Window.

Inputs:

* 0-6 options: T, num, max, min, start, state
* Inputs must be given in the above order. For example, if you want to define parameter “max”, the two previous parameters must also be defined.
* Parameters that are not defined will have the default value, as shown in Figure 1.

Output: Six lines in the Command Window, stating the current value of the parameters

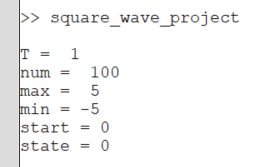


Figure 3. Output of square\_wave\_project with default values

### Function 2: get\_default\_options

Usage: get\_default\_options(varargin)

Description: This function is used to get the defined parameters and fill the rest with default values.

Input: A two dimensional array with all defined parameters stored in array(1)

Output: An array with six elements containing in order T, num, max, min, start, state

### Function 3: make\_square\_wave

Usage: make\_square\_wave (T, num, max, min, start, state, t)

Description: This function uses the formulas developed above and calculates the sum of all harmonics for each time instance in array “t”.

Input:

* Six outputs from function [get\_default\_options](#_Function_2:_get_default_options)
* One-dimensional array t, whose elements are time instances to be calculated and later drawn. The time instance starts with the parameter “start” and ends after 3 periods. Each period is represented using 500 points.
* The number of points each period (500) and the number of periods to compute and display (3) can only be specified by changing the constants in the source code.

Output:

* One-dimensional array f, containing the values of the generated square wave
* One-dimensional array aN, containing the values of all
* One-dimensional array bN, containing the values of all
* Two-dimensional array cos\_component, containing the value of all generated cosine waves
* Two-dimensional array sin\_component, containing the value of all generated sine waves

### Function 4: visualization

Usage: visualization(f, t, aN, bN, freq, cos\_component, sin\_component)

Description: This function takes the outputs from function [make\_square\_wave](#_Function_3:_make_square_wave) and plot the square wave in time, together with the most significant sine and cosine components. It also plots the amplitude-frequency bar graphs of the components.

Input:

* f, aN, bN, cos\_component, sin\_component from function [make\_square\_wave](#_Function_3:_make_square_wave)
* Array t, as described in the definition of the function [make\_square\_wave](#_Function_3:_make_square_wave).
* freq, the frequency of the square wave, equal to

Output:

* The plot of the square wave in time
* The plot of the first sine component in time
* The plot of the first cosine component in time
* The bar graph of the amplitude of the sine components in frequency domain
* The bar graph of the amplitude of the cosine components in frequency domain

# Result

Command: square\_wave\_project

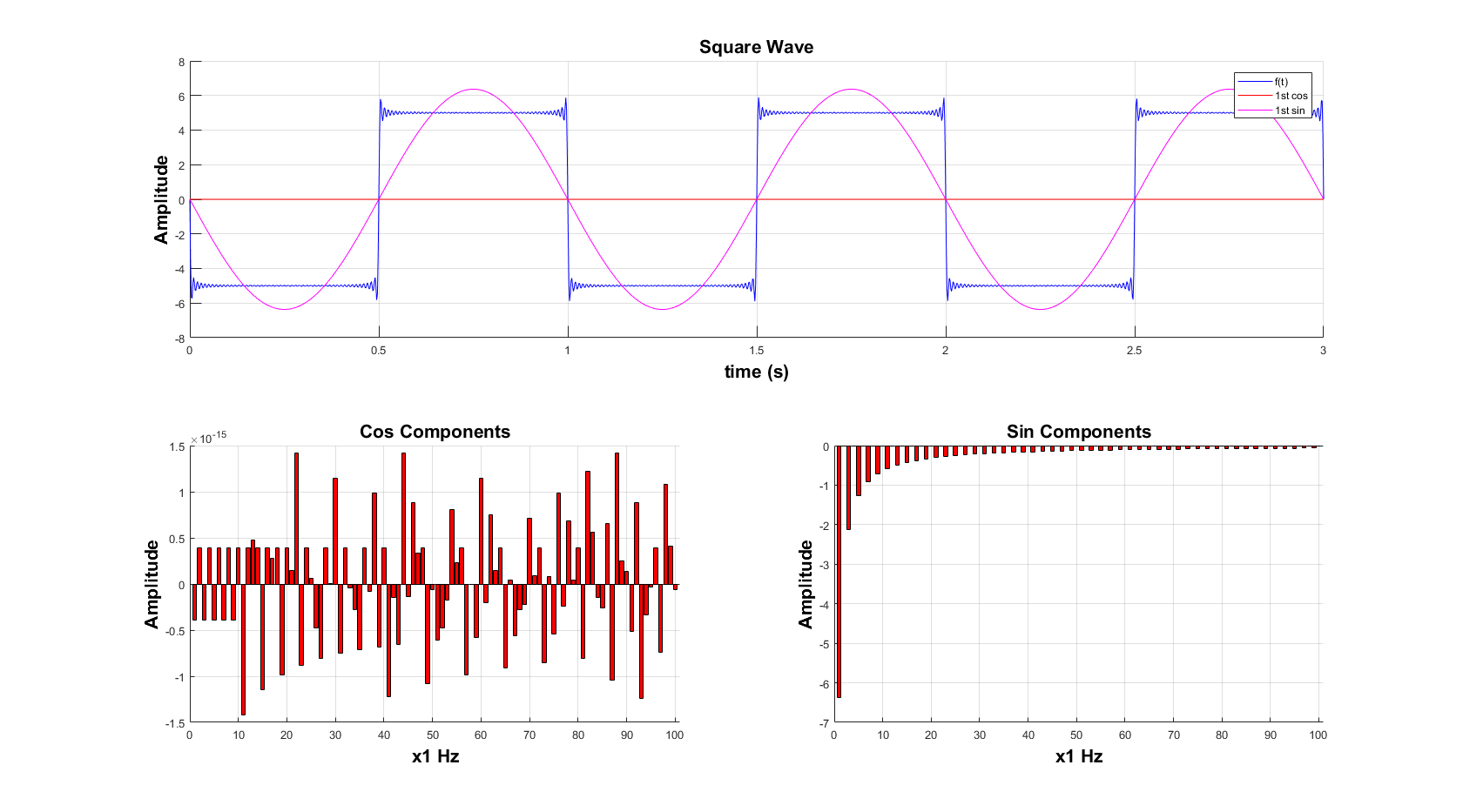
**

Figure 4. Screenshot of square\_wave\_project

Values of the options:

|  |  |  |
| --- | --- | --- |
| Option | Value | Note |
| T | 1 | Default value |
| num | 100 | Default value |
| max | 5 | Default value |
| min | -5 | Default value |
| start | 0 | Default value |
| state | 0 | Default value |

Figure 5. Options’ values of square\_wave\_project

As you may notice, there is a small value for the amplitude of the cosine components. As derived before, our formula for is:

So, I suggest that the small value we had here in our calculation is the rounding error of the floating-point computation.

Command: square\_wave\_project(5,500,10,-3,2)

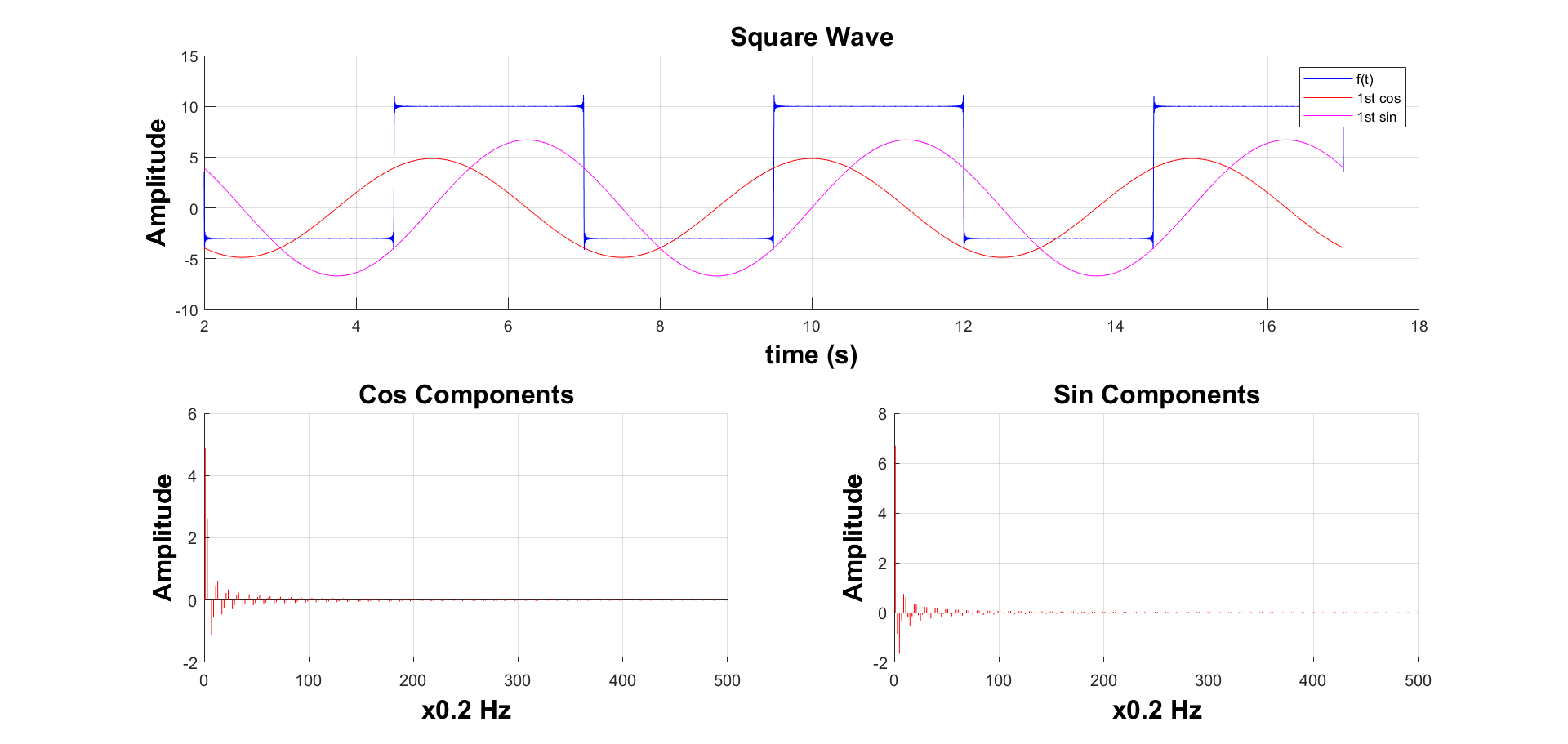


Figure 6. Screenshot of square\_wave\_project(5,500,10,-3,2)

Values of the options:

|  |  |  |
| --- | --- | --- |
| Option | Value | Note |
| T | 5 |  |
| num | 500 |  |
| max | 10 |  |
| min | -3 |  |
| start | 2 |  |
| state | 0 | Default value |

Figure 7. Options’ values of square\_wave\_project(5,500,10,-3,2)

Command: square\_wave\_project(3,50,300,-300,200,1)

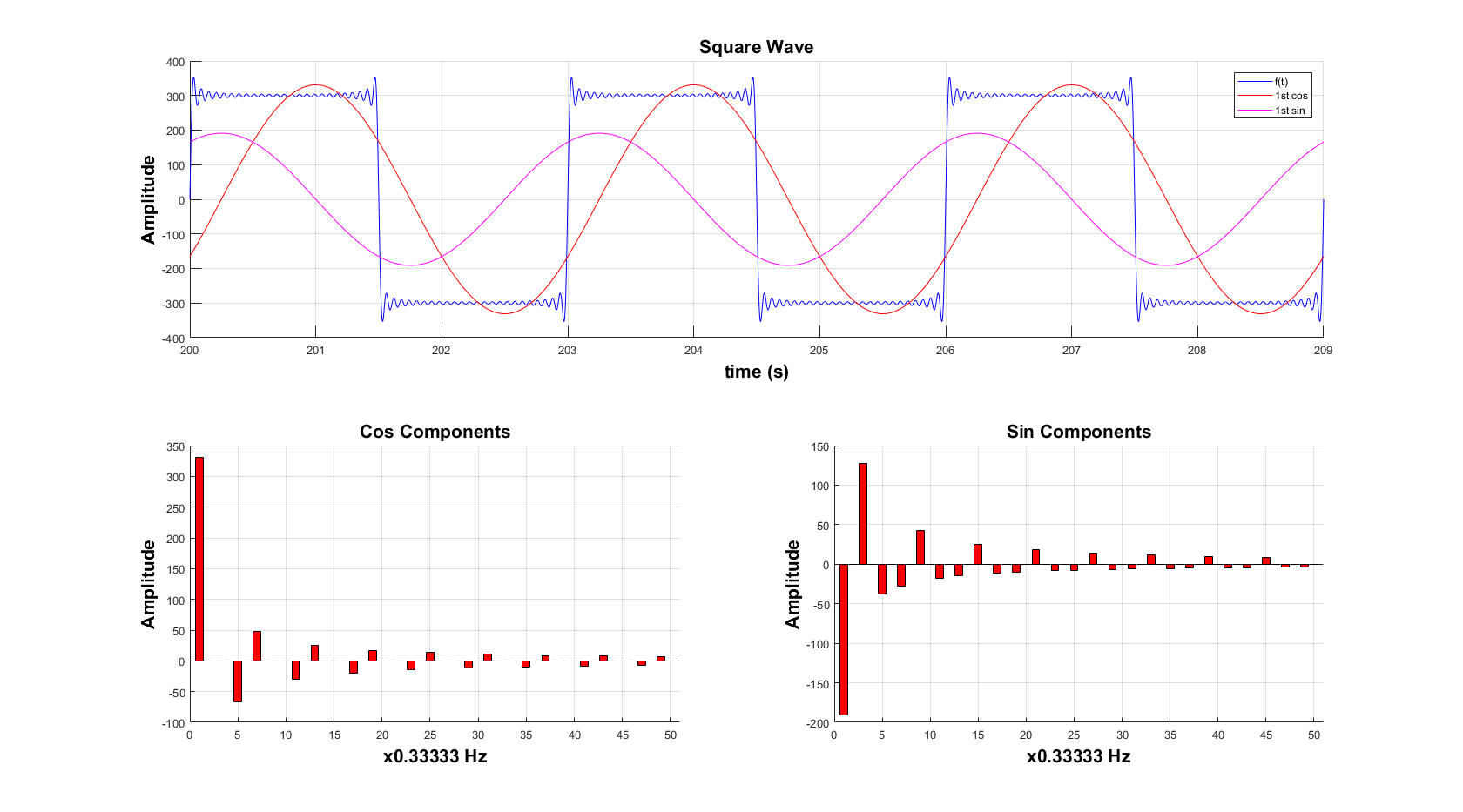


Figure 8. Screenshot of square\_wave\_project(3,50,300,-300,200,1)

Values of the options:

|  |  |  |
| --- | --- | --- |
| Option | Value | Note |
| T | 3 |  |
| num | 50 |  |
| max | 300 |  |
| min | -300 |  |
| start | 200 |  |
| state | 1 |  |

Figure 9. Options’ values of square\_wave\_project(3,50,300,-300,200,1)

As one can observe, the square wave looks better with the increase in the number of addends to compute. If there is enough computation power, for better results, one should increase that number along with the number of points to compute each period as mentioned in the function [make\_square\_wave](#_Function_3:_make_square_wave).

Due to the fact that I have been using both Octave (free software) and Matlab (trial version) in this project. I find that Matlab runs the same code of mine a lot faster than Octave. I can increase the number of addends up to 1000 with no problem of having the long computation time. Additionally, I can save the plot as an png image in Matlab using the save icon in the plot, rather than adding some lines of code as in Octave.

# References

The derivation of Fourier series

<https://www.khanacademy.org/science/electrical-engineering/ee-signals>

The code for using default parameters:

<https://blogs.mathworks.com/loren/2009/05/05/nice-way-to-set-function-defaults/>