Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Number	ELE 532
Course Title	Signals and Systems I
Semester/Year	Fall 2025
Instructor	Beheshti, Soosan
TA Name	Punya Cheema

Lab/Tutorial Report No.	01
Report Title	Working with MATLAB, Visualization of Signals

Submission Date	2025.09.28
Due Date	September 28, 2025

Student Name	Student ID	Signature
Vinci Fajardo	501239903	VF
Keunhyeok Choi	500958125	KH

PROBLEM A:

A.1:

Figure 1.46:

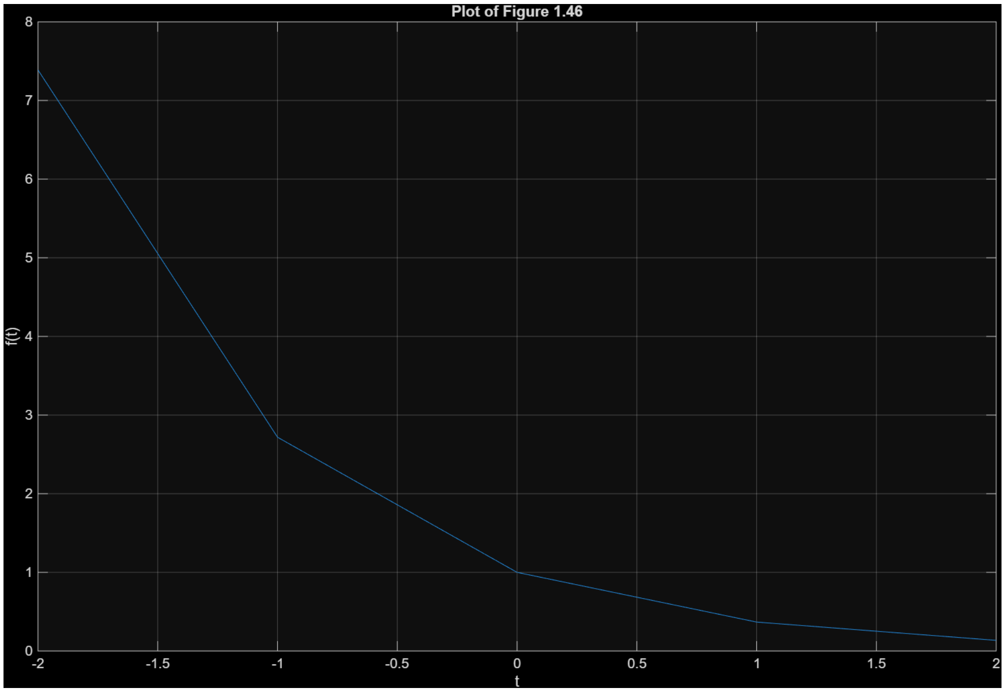
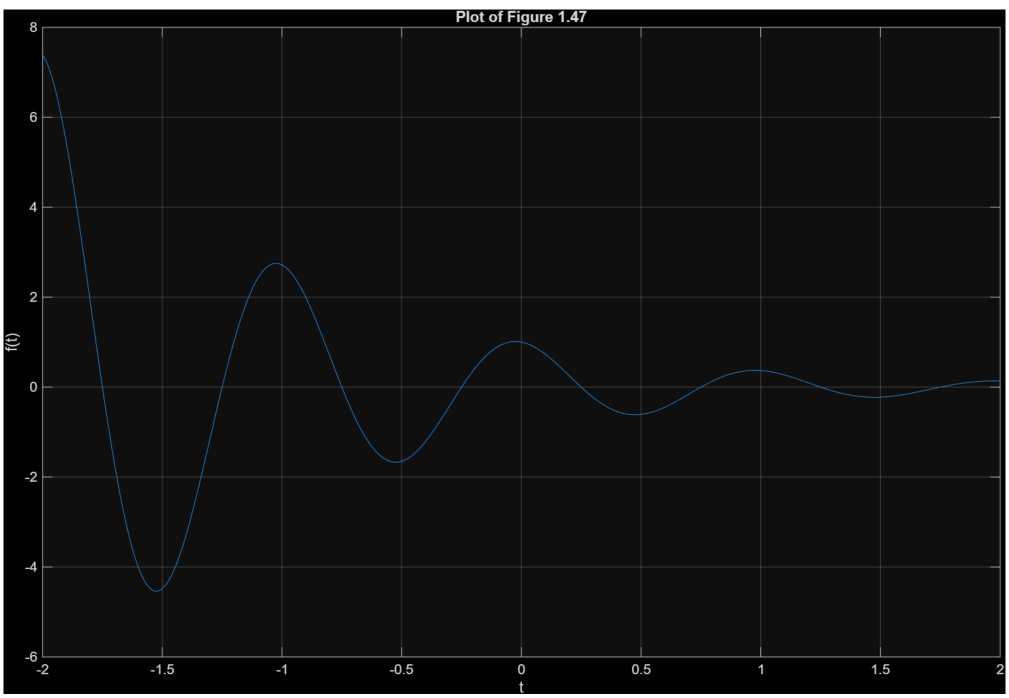
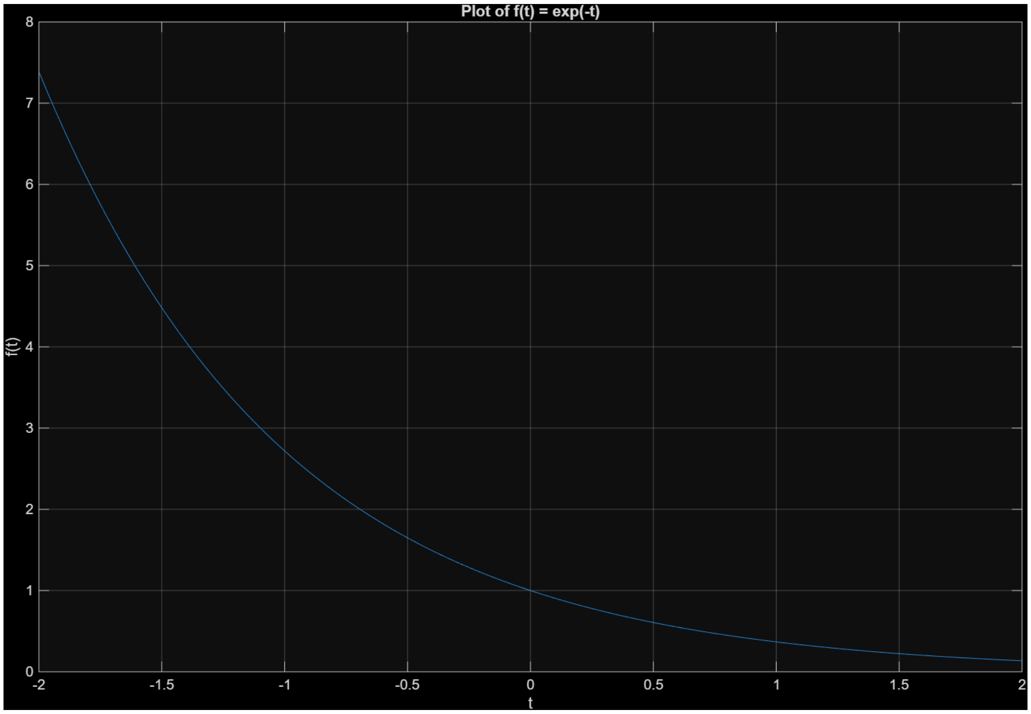


Figure 1.47:



A.2:

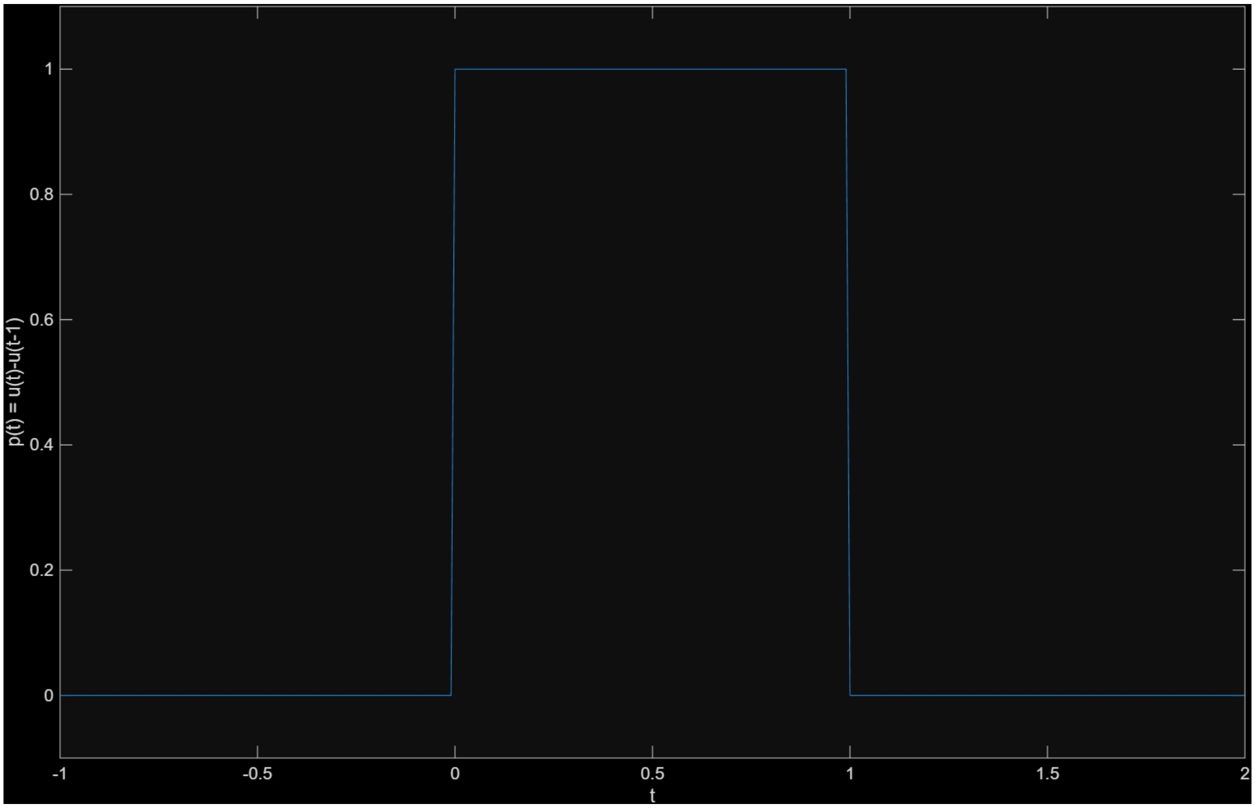


For Problem A.3: Comparison of Plots

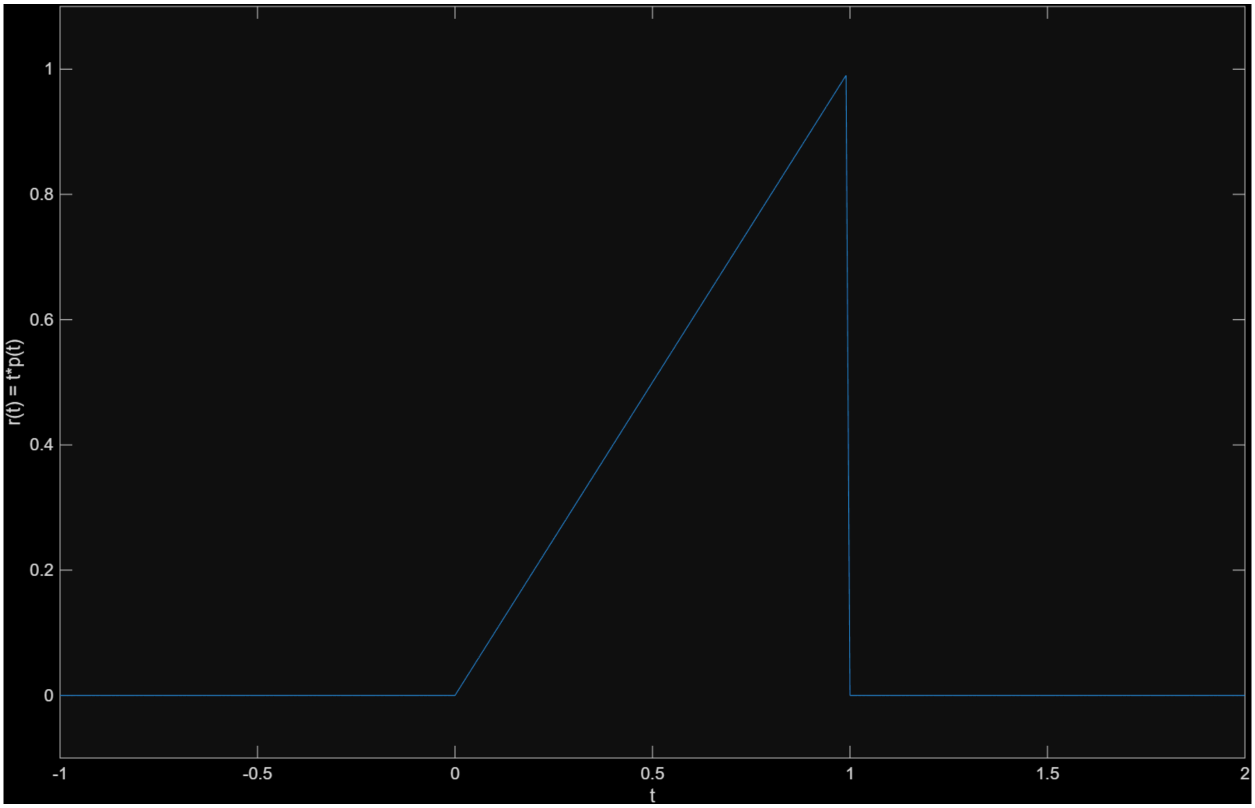
The plot generated in Problem A.1 appears as a smooth, continuous function because the time vector t = [-2:0.01:2] has a very high resolution with many points. In contrast, the plot in Problem A.2, which uses the vector t = [-2:1:2] , only shows five discrete points. This comparison demonstrates that the visual representation of a function in MATLAB is highly dependent on the resolution of its independent variable vector.

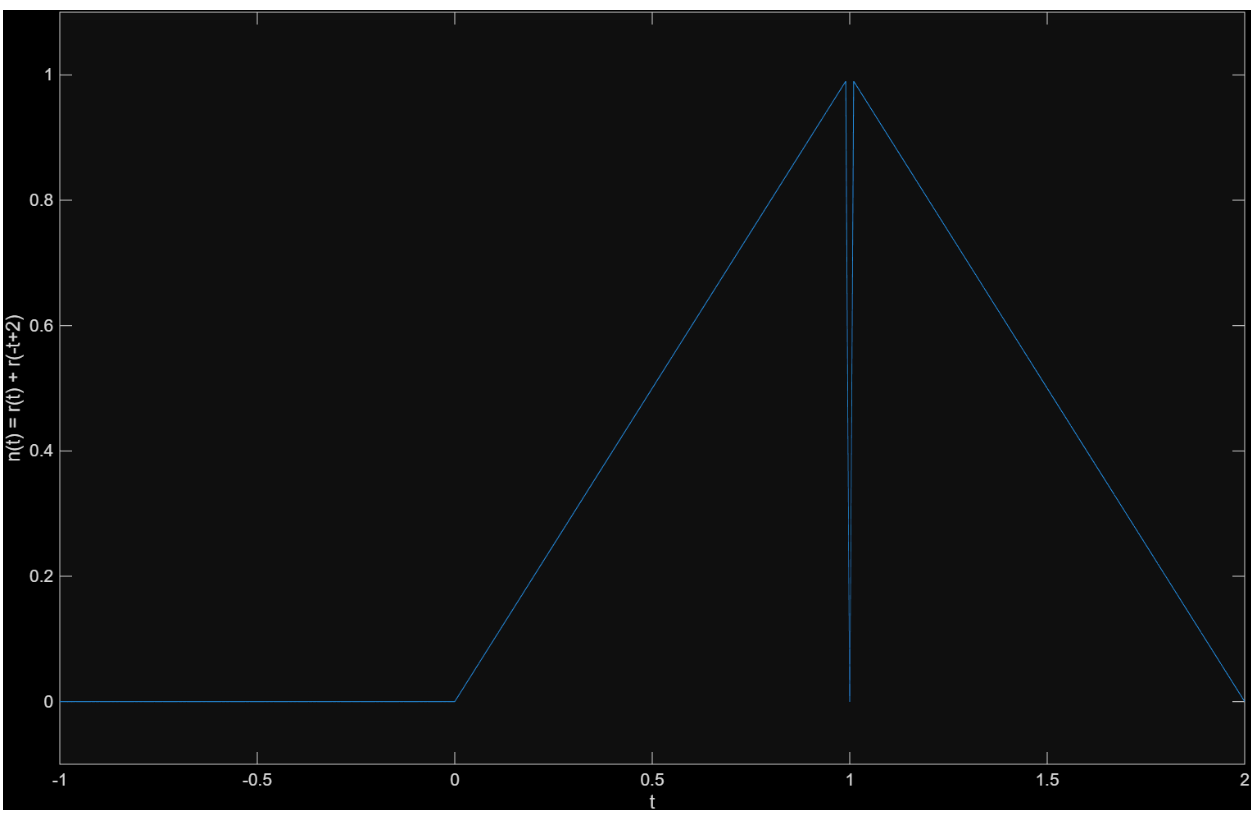
PROBLEM B:

B.1

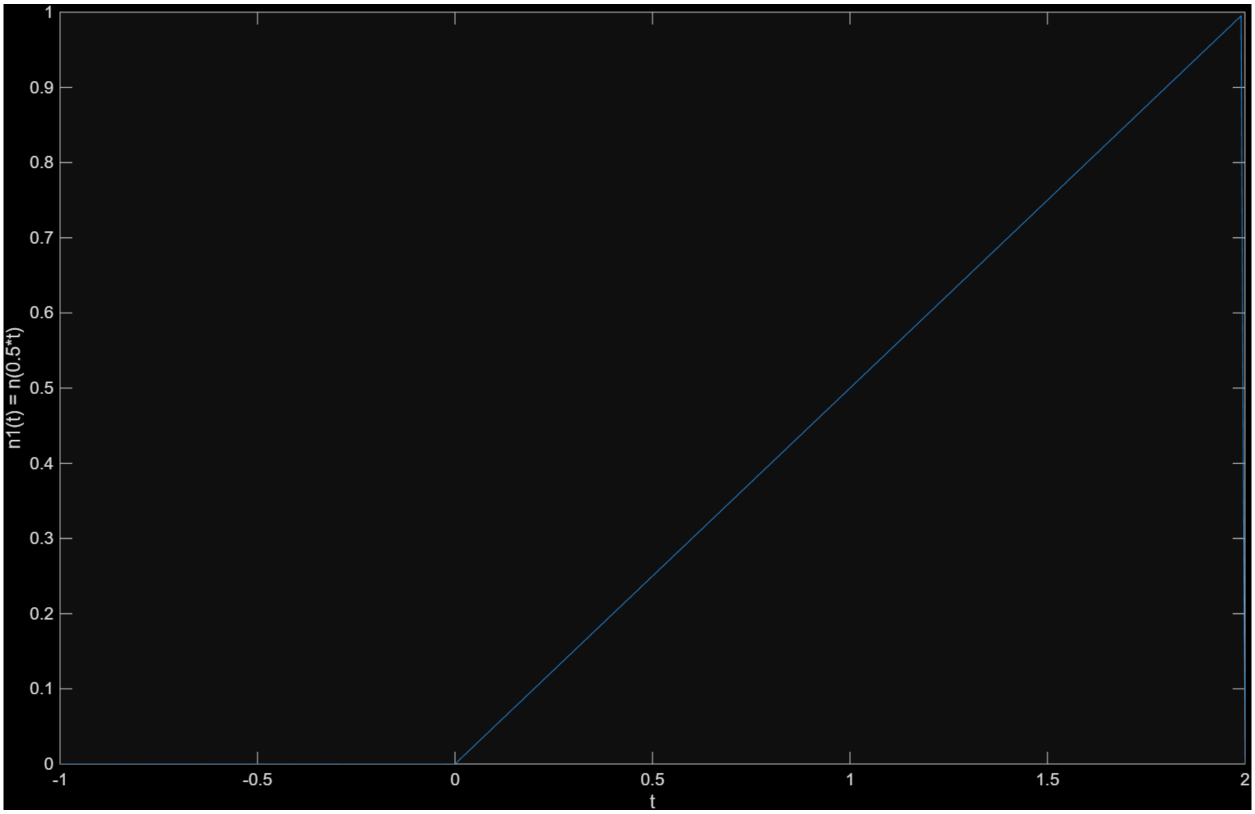


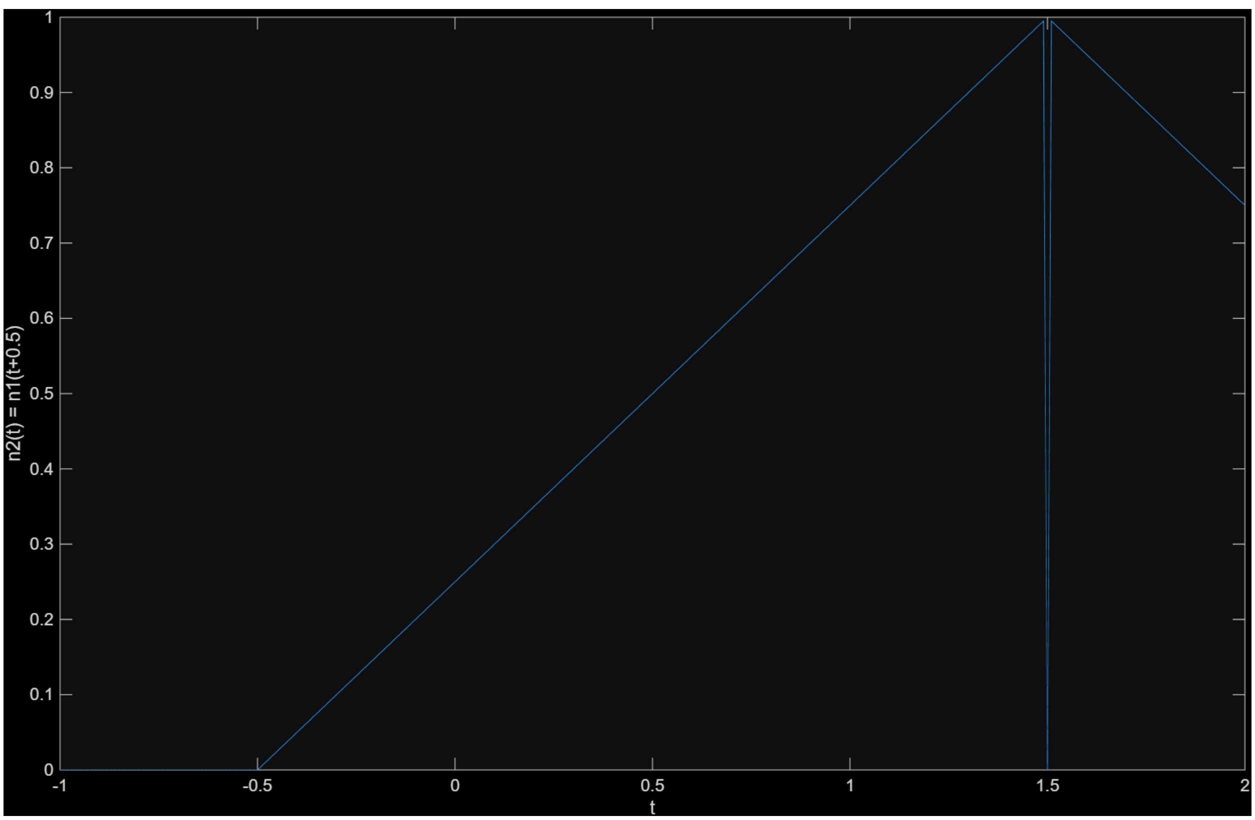
B.2


r(t) = t*p(t)
1
0.8
0.6
0.4
0.2
0
-1
-0.5
0
0.5
1
1.5
2
t

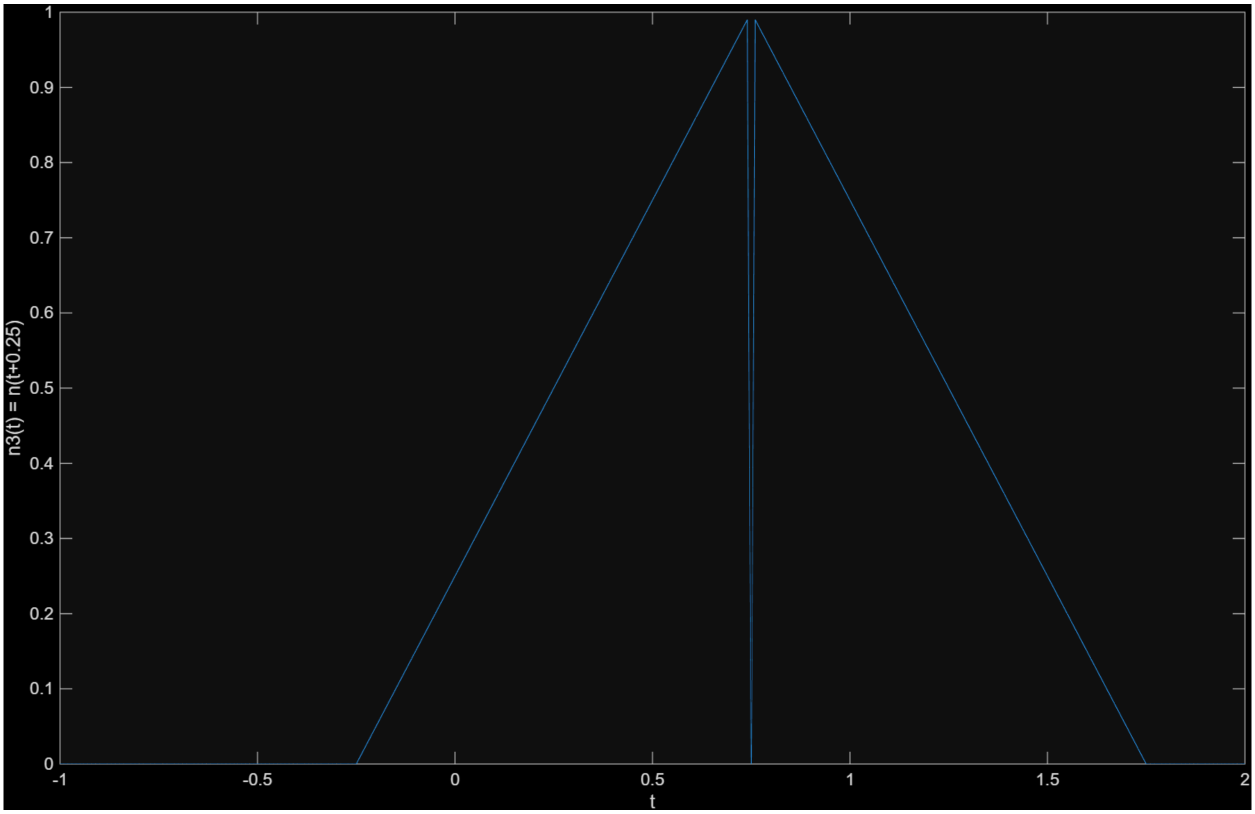

n(t) = r(t) + r(-t+2)
1
0.8
0.6
0.4
0.2
0
-1
-0.5
0
0.5
1
1.5
2
t

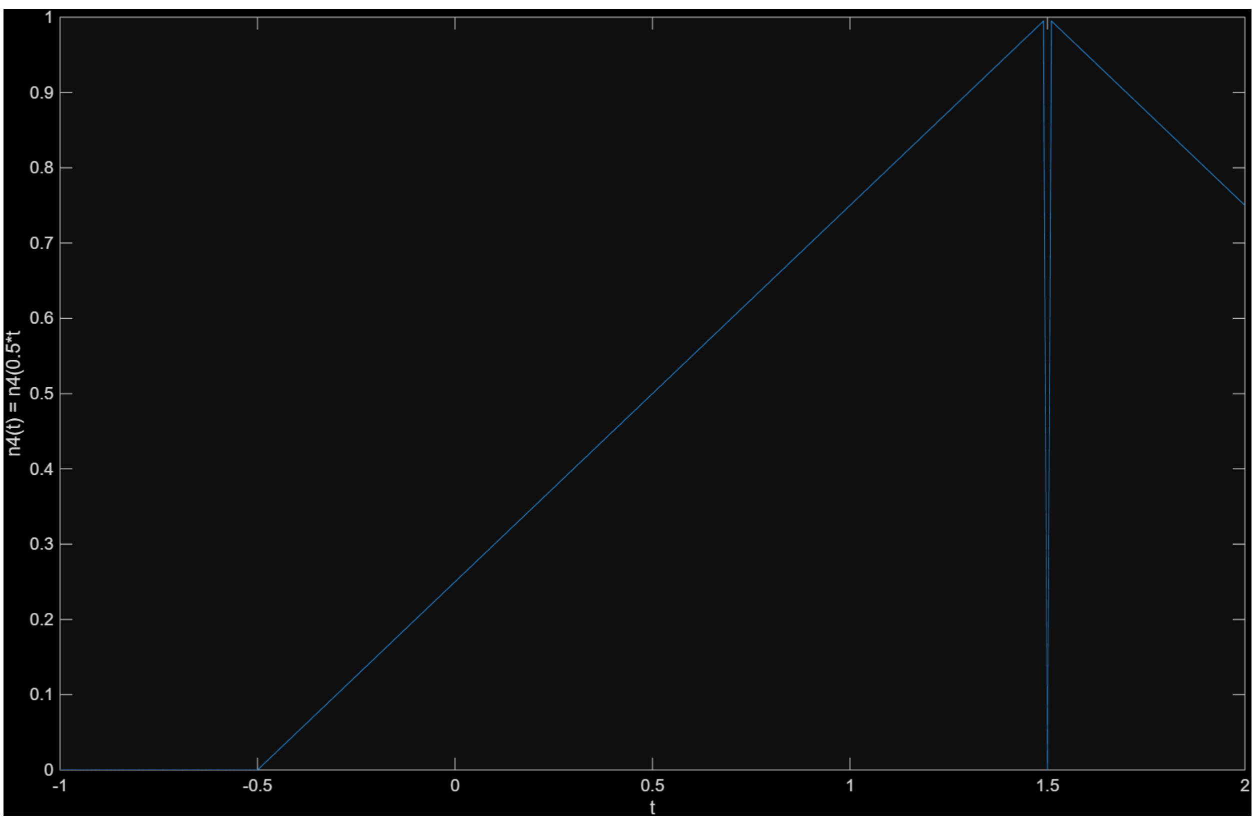
B.3





B.4


n3(t) = n(t+0.25)
t
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1
1.5
2


n4(t) = n4(0.5*t
t
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1
1.5
2

B.5

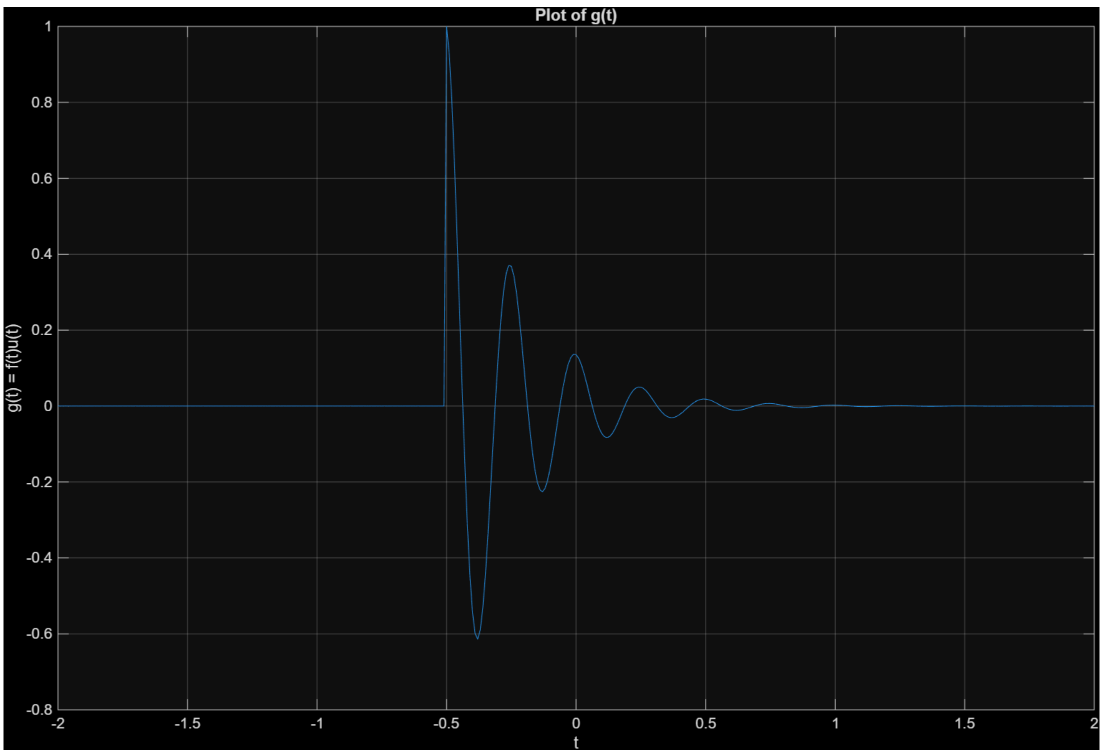
The plots for n2(t) and n4(t) are identical because, despite the different order of operations, they both simplify to the same mathematical expression.

- **For n2(t)**, the process is time scaling followed by a time shift: $n(t) \rightarrow n(0.5t) \rightarrow n(0.5(t+0.5)) = n(0.5t+0.25)$
- **For n4(t)**, the process is a time shift followed by time scaling: $n(t) \rightarrow n(t+0.25) \rightarrow n((0.5t)+0.25) = n(0.5t+0.25)$

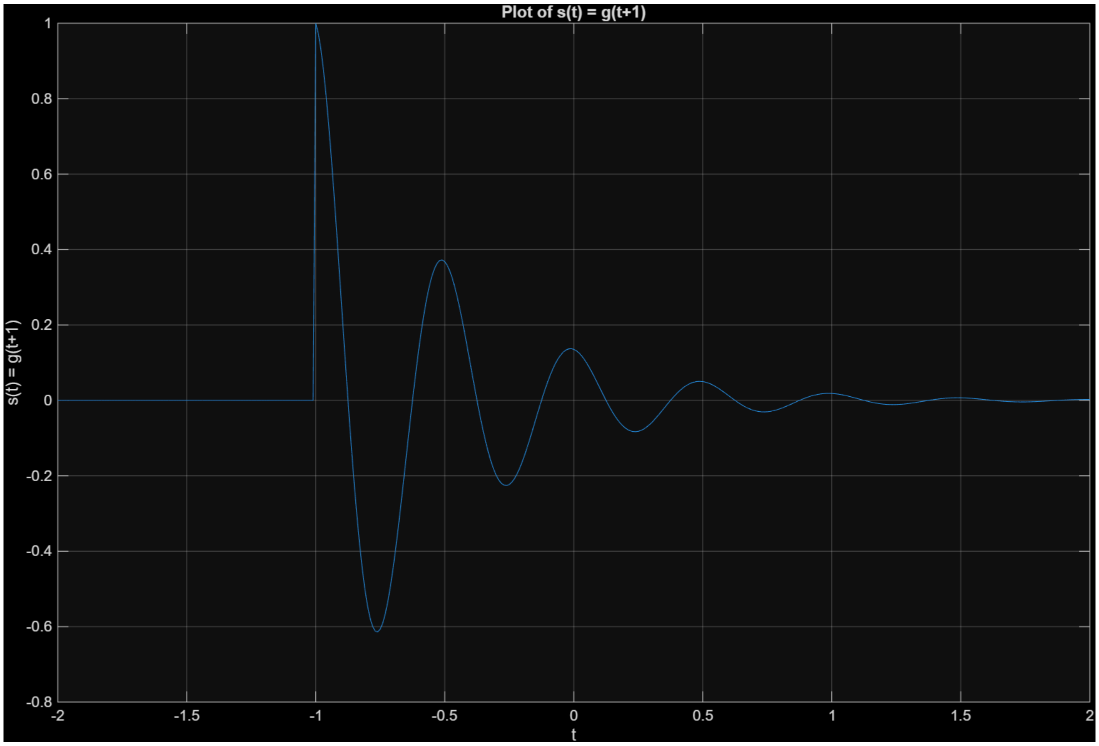
Since both operations result in the same final function, their plots are identical. This exercise highlights the importance of the order of operations in signal transformation.

PROBLEM C:

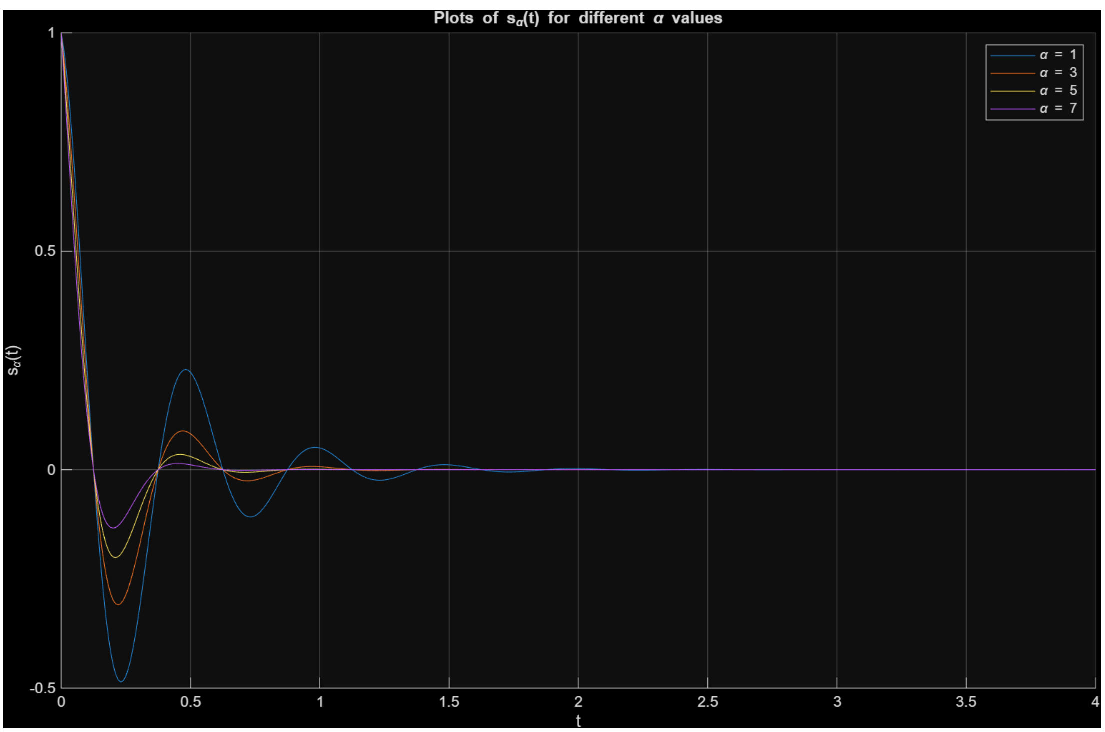
C.1:



C.2:


Plot of s(t) = g(t+1)
s(t) = g(t+1)
t
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

C.3:


Plots of $s_\alpha(t)$ for different α values
α = 1
α = 3
α = 5
α = 7
$s_\alpha(t)$
t
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4

C.4:
Using a built-in command from MatLab, the size of the matrix can be determined. After running this code, the size of the matrix is revealed to be 4 by 401.

PROBLEM D:

For Problem D.1: Operation Results and Descriptions

(a) A(:) *This operation reshapes the matrix A into a single column vector by stacking its columns.*

```
ans =
    0.5377
    1.8339
   -2.2588
    ...
```

(b) A([2 4 7]) *This operation extracts the 2nd, 4th, and 7th elements from the column vector created by linear indexing.*

```
ans =
    1.8339
    0.3426
   -0.4336
```

(c) [A >= 0.2] *This operation returns a logical matrix of the same size as A, with* `1` *(true) where the condition is met and* `0` *(false) otherwise.*

```
ans =
  5×4 logical array
   1   0   0   0
   1   0   1   0
   0   1   1   0
   1   1   0   1
   1   0   1   1
```

(d) A([A >= 0.2]) *This logical indexing operation extracts all elements from A that are greater than or equal to 0.2 and displays them as a column vector.*

```
ans =
   0.5377
   1.8339
   ...
```

(e) A([A >= 0.2]) = 0 *This operation finds all elements in A greater than or equal to 0.2 and sets their value to 0.*

```
A =
0         -1.3077   -1.3499    -0.2050
0         -0.4336   0          -0.1241
-2.2588   0         0          -0.0631
0         0         -0.0631    0
0         -0.4447   0          0
```

For Problem D.2: Execution Time Conclusion

After the `fprintf` outputs for D.2, add a concluding sentence:

The results clearly show that using MATLAB's logical indexing (~`0.0001` seconds) is significantly faster than using nested for-loops (~`0.002` seconds). This performance difference is because MATLAB is highly optimized for vectorized and matrix operations.