calculateFromFreq.m

```
sequence = input("Enter the sequence in frequency domain: ");

lengthOfSequence = length(sequence);
IDFT = idft(sequence);

round(IDFT, 4)

figure;
stem(linspace(0, lengthOfSequence - 1, lengthOfSequence),
abs(IDFT), "lineWidth", 1.5);
set(get(gca, 'XLabel'), 'String', 'n \rightarrow');
set(get(gca, 'YLabel'), 'String', 'Amplitude in time domain');
set(get(gca, 'Title'), 'String', 'Inverse DFT from Freq domain to
time domain.');
grid on;

for index = 0:lengthOfSequence - 1
text(index, IDFT(index + 1), strcat('\rightarrow',
num2str(abs(round(IDFT(index + 1), 2)))));
```

```
end

axisData = axis;
padding = 0.1;
axisLength = axisData(2) - axisData(1);
axisHeight = axisData(4) - axisData(3);
axis([axisData(1) - padding * axisLength axisData(2) + padding * axisLength axisData(3) - padding * axisHeight axisData(4) + padding * axisHeight]);
```

calculateFromTime.m

```
clear;
clc;
close all;

sequence = input("Enter the Sequence in time domain: ");

lengthOfSequence = length(sequence);
```

```
DFT = dft(sequence);
[DTFT pointLength] = dtft(sequence);
"Calculated using manually coded function"
round(DFT, 3)

"Calculated using builtin coded function"
round(fft(sequence).', 3)

subplot(2, 1, 1);
plot(linspace(0, lengthOfSequence, pointLength), abs(DTFT),
"lineWidth", 1);
hold on;
grid on;
stem(linspace(0, lengthOfSequence - 1, lengthOfSequence),
abs(DFT), "lineWidth", 1.5);
hold off;
setPlotAttributes('n \rightarrow \omega \rightarrow', 'DTFT/DFT
Amplitude', 'Amplitude of DTFT & DFT from a time domain signal')

for index = 0:lengthOfSequence - 1
```

```
text(index, abs(DFT(index + 1)), strcat('\rightarrow',
num2str(round(abs(DFT(index + 1)), 2))));
end

setAxisLimits(axis);

phase = angle(round(DFT, 6)) * 180  /pi;

subplot(2, 1, 2);
stem(linspace(0, lengthOfSequence - 1, lengthOfSequence), phase,
"lineWidth", 1.5);
grid on;
setPlotAttributes('n \rightarrow', 'Angle in Degrees (°)', 'Angle of
DFT from time domain Signal')

for index = 0:lengthOfSequence - 1
text(index, phase(index + 1), strcat('\rightarrow',
num2str(phase(index + 1)), '°'));
end

setAxisLimits(axis);
```

dft.m

```
function DFT = dft(sequence)

sequenceLength = length(sequence);
twiddleFactorMatrix = twiddleFactor(sequenceLength);
DFT = twiddleFactorMatrix * sequence.';

end
```

dtft.m

```
function [DTFT dtftPointLength] = dtft(sequence)
sequenceLength = length(sequence);
dtftPointLength = 1000;
DTFT = dft([sequence zeros(1, dtftPointLength -
sequenceLength)]);
end
```

idft.m

```
function IDFT = idft(sequence)

IDFT = conj(dft(conj(sequence.'))) /length(sequence);

end
```

setAxisLimits.m

```
function setAxisLimits(axisData)
padding = 0.1; % Relative to the overall output
axisLength = axisData(2) - axisData(1);
axisHeight = axisData(4) - axisData(3);
axis([axisData(1) - padding * axisLength axisData(2) + padding *
axisLength axisData(3) - padding * axisHeight axisData(4) +
padding * axisHeight]);
end
```

setPlotAttributes.m

```
function setPlotAttributes(xAxisLabel, yAxisLabel, plotTitle)
```

```
set(get(gca, 'XLabel'), 'String', xAxisLabel);
set(get(gca, 'YLabel'), 'String', yAxisLabel);
set(get(gca, 'Title'), 'String', plotTitle);
end
```

twiddleFactor.m

```
function twiddleMatrix = twiddleFactor(sequenceLenght,
needMatrix)

arguments
sequenceLenght
needMatrix = 1;
end

twiddleMatrix = ones([sequenceLenght sequenceLenght]);
theta = 2 * pi  /sequenceLenght;

for index1 = 2:sequenceLenght

for index2 = 2:sequenceLenght
```

```
twiddleMatrix(index1, index2) = cos(theta * (index2 - 1) * (index1
- 1)) - i * sin(theta * (index2 - 1) * (index1 - 1));
end

end

if ~needMatrix
twiddleMatrix = twiddleMatrix(:, 2);
end

end
```