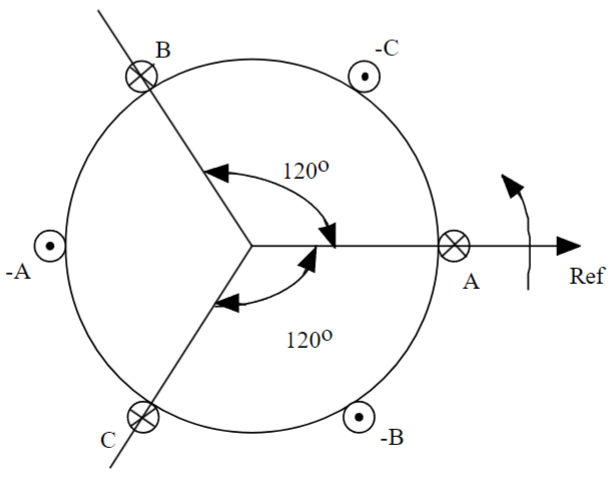
ECEN 611 Homework 2: Sinusoidal

Shuxuan Chen | UIN: 132006082 | Fall 2024

*Last Updated: Sep 22*

Table of Contents

[Problem 1](#MW_H_4A3FD973)   
 [(a) Winding Function Diagrams for Windings A, B, and C](#MW_H_B9DB3A63)   
 [General Settings](#MW_H_8C99DF8F)   
 [Counting Functions for Windings A, B, and C](#MW_H_C9AFDCE8)   
 [Winding Functions for Windings A, B, and C](#MW_H_4AB1FE44)   
 [Plot Counting Functions and Winding Functions A, B, and C](#MW_H_564D14E3)   
 [(b) MMF of Windings A, B, and C Supplied by Three Phase Sinusoidal Currents](#MW_H_5C3769CE)   
 [Three Phase Sinusoidal Currents](#MW_H_3742C372)   
 [Three Phase Currents at Angle of Interest](#MW_H_4782FEB6)   
 [Three Phase MMFs at Angles of Interest](#MW_H_E880D25D)   
 [(c) Total MMF at Angles of Interest](#MW_H_F4BE4604)   
[Problem 2](#MW_H_65280B52)   
 [(d) Fourier Series Analysis](#MW_H_4A9044CF)   
 [Winding Functions Represented by Fourier Series](#MW_H_77276C02)   
 [Sinusoidal Currents Represented by Fourier Series](#MW_H_B9AF31C1)   
 [Phase MMF Represented by Fourier Series at Angles of Interest](#MW_H_D74B67B5)   
 [Total MMF Represented by Fourier Series at Angles of Interest](#MW_H_5F8DFCEB)   
 [Table](#MW_H_0CFFF6FC)



clearvars

clc

Numeric computation is always recommended when it can solve the problem with acceptable precision.

# Problem 1

## (a) Winding Function Diagrams for Windings A, B, and C

Draw the winding functions for windings A, B, and C on the same sheet of paper.

### General Settings

% Define the spatial angle from 0 to 360 degrees

T = 2\*pi;

T\_degree = rad2deg(T)

T\_degree = 360

numOfPoints = 5000; % This affects precision of Fourier analysis

theta = linspace(0, T, numOfPoints);

theta\_degrees = linspace(0, T\_degree, numOfPoints)

theta\_degrees = *1×5000*

0 0.0720 0.1440 0.2160 0.2881 0.3601 0.4321 ⋯

### Counting Functions for Windings A, B, and C

% Initialize counting function to zero arrays

nA = zeros(size(theta));

nB = zeros(size(theta));

nC = zeros(size(theta));

% Define nA

nA(mod(theta, T) >= 0 & mod(theta, T) < T/2) = 1;

nA(mod(theta, T) >= T/2 & mod(theta, T) <= T) = 0;

**Explanation:**

* **mod(theta, T)**: This ensures that the function repeats every T. By applying mod(theta, T), you're effectively folding the waveform to repeat from T to 2T.
* **First Condition**: For the interval 0≤θ≤T/2, the function returns 1.
* **Second Condition**: For the interval T/2<θ≤T, the function returns nA\_pi - 1.

% Define nB

% nB(theta >= 0 & theta < 120) = 0;

% nB(theta >= 120 & theta < 300) = 1;

% nB(theta >= 300 & theta <= 360) = 0;

%

% % Define nC

% nC(theta >= 0 & theta < 60) = 1;

% nC(theta >= 60 & theta < 240) = 0;

% nC(theta >= 240 & theta <= 360) = 1;

% Define phase shift for iB and iC

phase\_shift = 2\*pi/3;

% Define iB as phase-shifted iA

nB = circshift(nA, [0, round(phase\_shift/(2\*pi) \* length(theta))]);

% Define iC as phase-shifted iA by 2\*phase\_shift

nC = circshift(nA, [0, round(2\*phase\_shift/(2\*pi) \* length(theta))]);

### Winding Functions for Windings A, B, and C

% Define NA, NB, and NC

NA = nA - mean(nA);

NB = nB - mean(nB);

NC = nC - mean(nC);

### Plot Counting Functions and Winding Functions A, B, and C

figure

tiledlayout(3,1)

titleHandle = sgtitle("Counting Function and Winding Function");

titleHandle.FontSize = 16;

% First tile: nA and NA

nexttile;

plot(theta\_degrees, nA, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NA, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase A Winding");

legend('nA', 'NA');

xticks(0:30:360);

% Second tile: nB and NB

nexttile;

plot(theta\_degrees, nB, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NB, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase B Winding");

legend('nB', 'NB');

xticks(0:30:360);

% Third tile: nC and NC

nexttile;

plot(theta\_degrees, nC, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NC, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

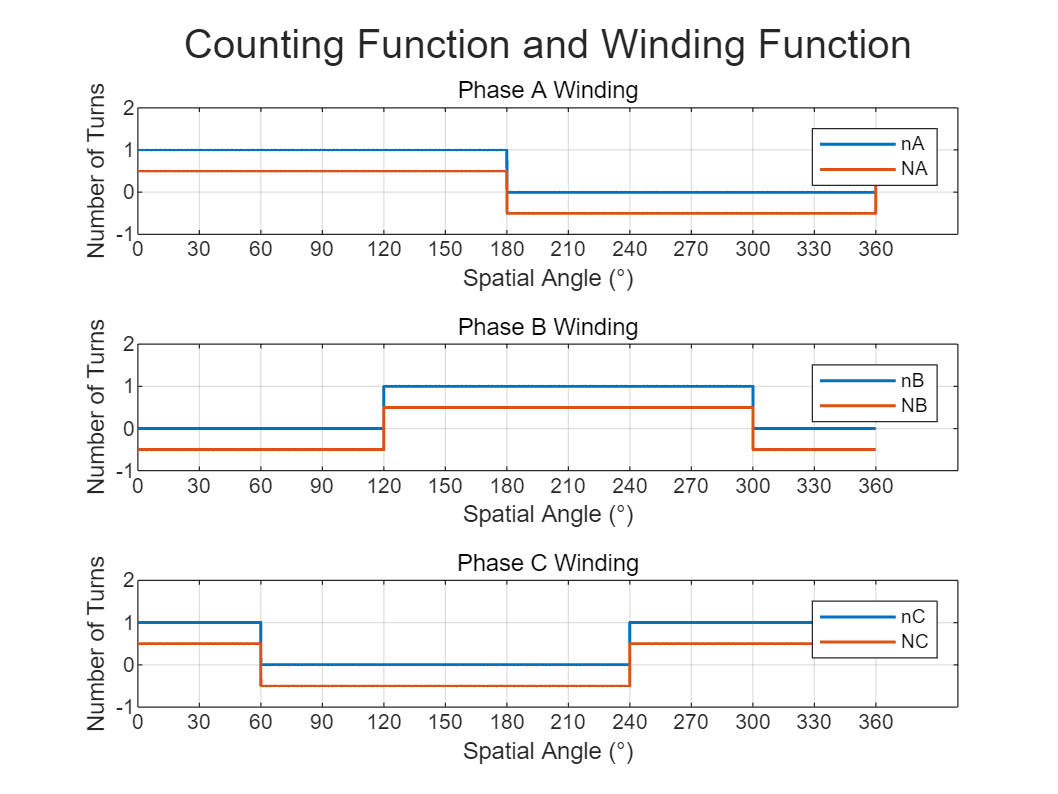
xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase C Winding");

legend('nC', 'NC');

xticks(0:30:360);



% Counting functions √

% Winding functions √

## (b) MMF of Windings A, B, and C Supplied by Three Phase Sinusoidal Currents

Draw the MMF of each phase if the windings are supplied by three phase sinusoidal currents at ωt = 0°, 30° , 60° , 90° , 120°, ...

### 

### Three Phase Sinusoidal Currents

wt = linspace(0, T, numOfPoints);

wt\_degrees = rad2deg(wt);

[iA, iB, iC] = generateSinusoidalCurrents(wt);

% Generate sinusoidal currents for three phases

function [iA, iB, iC] = generateSinusoidalCurrents(wt)

iA = cos(wt);

iB = cos(wt - 2\*pi/3);

iC = cos(wt + 2\*pi/3);

% iA = sin(wt);

% iB = sin(wt - 2\*pi/3);

% iC = sin(wt + 2\*pi/3);

end

% Ensure wt\_degrees, iA, iB, and iC are defined

figure;

plot(wt\_degrees, iA, ...

wt\_degrees, iB, ...

wt\_degrees, iC, 'LineWidth', 1.2);

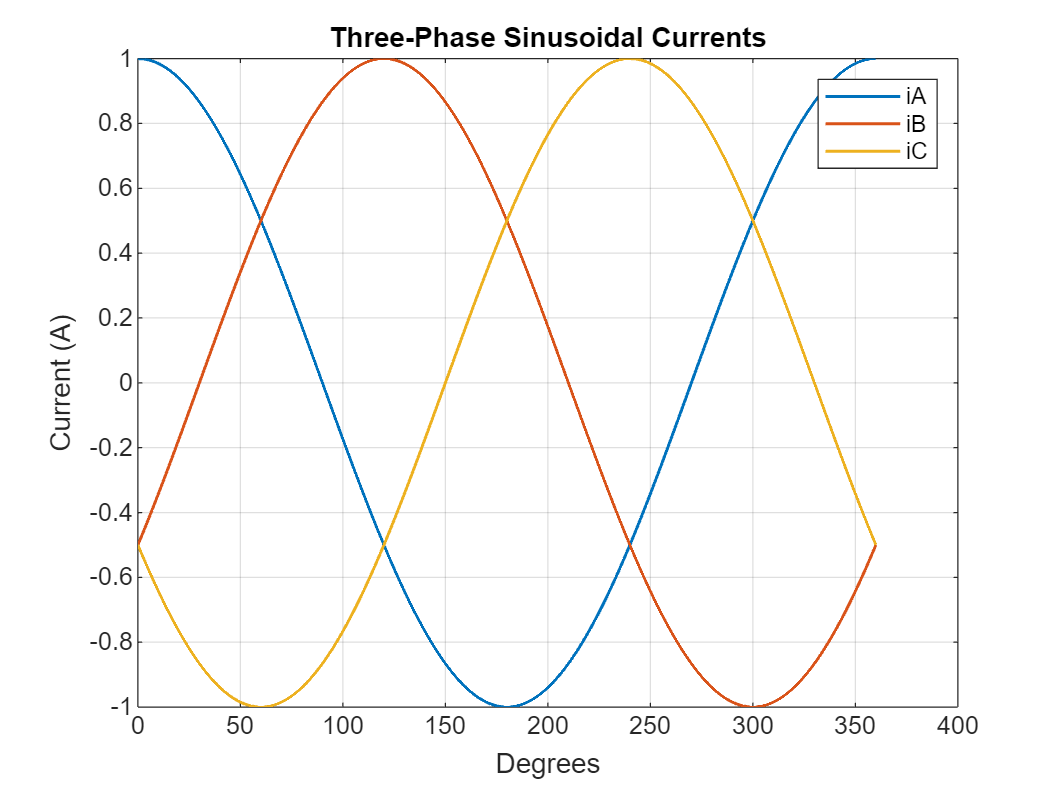
legend('iA', 'iB', 'iC');

title('Three-Phase Sinusoidal Currents');

xlabel('Degrees');

ylabel('Current (A)');

grid on;



### Three Phase Currents at Angle of Interest

angleOfInterest = 0:30:330;

% Pre-allocate the index array

indices = zeros(size(angleOfInterest));

% Loop through each angle of interest and find the closest match in wt\_degrees

for i = 1:length(angleOfInterest)

[~, idx] = min(abs(wt\_degrees - angleOfInterest(i))); % Find closest value

indices(i) = idx; % Store the index of the closest value

end

iA\_angleOfInterest = iA(indices);

iB\_angleOfInterest = iB(indices);

iC\_angleOfInterest = iC(indices);

% Ensure wt\_degrees, iA, iB, and iC are defined

figure;

plot(wt\_degrees(indices), iA\_angleOfInterest, ...

wt\_degrees(indices), iB\_angleOfInterest, ...

wt\_degrees(indices), iC\_angleOfInterest, 'LineWidth', 1.2);

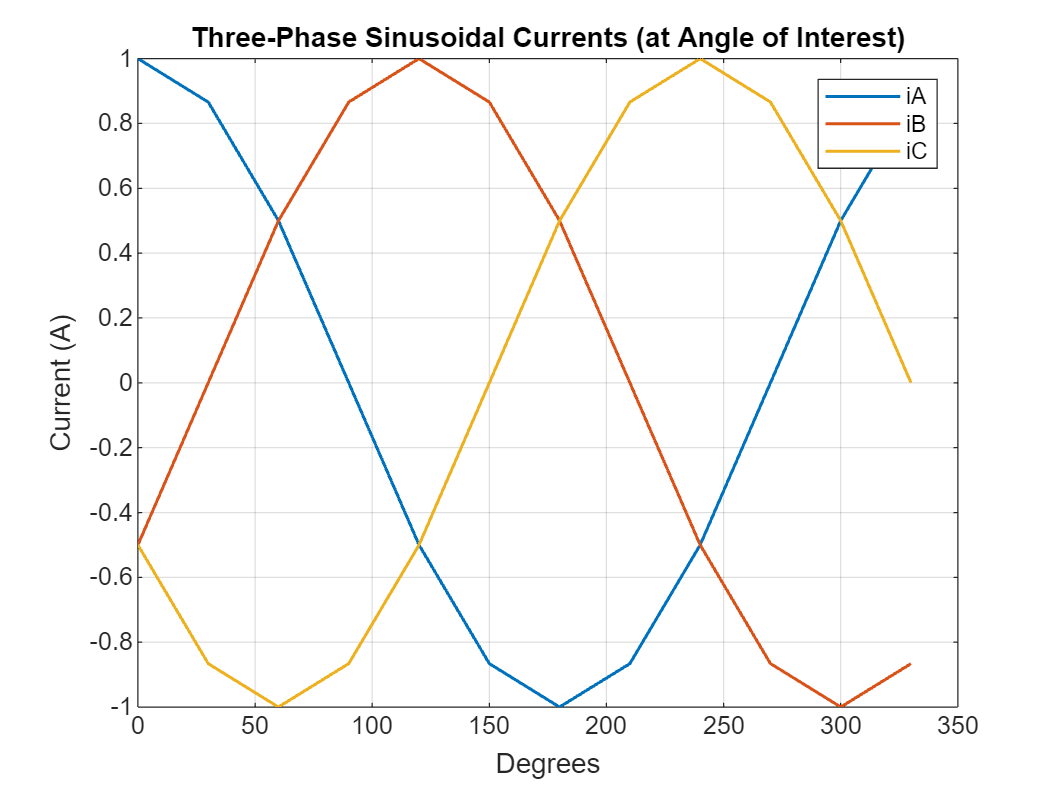
legend('iA', 'iB', 'iC');

title('Three-Phase Sinusoidal Currents (at Angle of Interest)');

xlabel('Degrees');

ylabel('Current (A)');

grid on;



### Three Phase MMFs at Angles of Interest

MMFA = NA .\* iA\_angleOfInterest';

MMFB = NB .\* iB\_angleOfInterest';

MMFC = NC .\* iC\_angleOfInterest';

for k = 1:length(angleOfInterest)

figure;

tiledlayout(3, 1);

% Common title with current wt in degrees

titleHandle = sgtitle(sprintf('MMF at ωt = %.0f°', angleOfInterest(k)));

titleHandle.FontSize = 13;

% Plot Phase A MMF

nexttile;

plot(wt\_degrees, MMFA(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase A Winding');

% Plot Phase B MMF

nexttile;

plot(wt\_degrees, MMFB(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase B Winding');

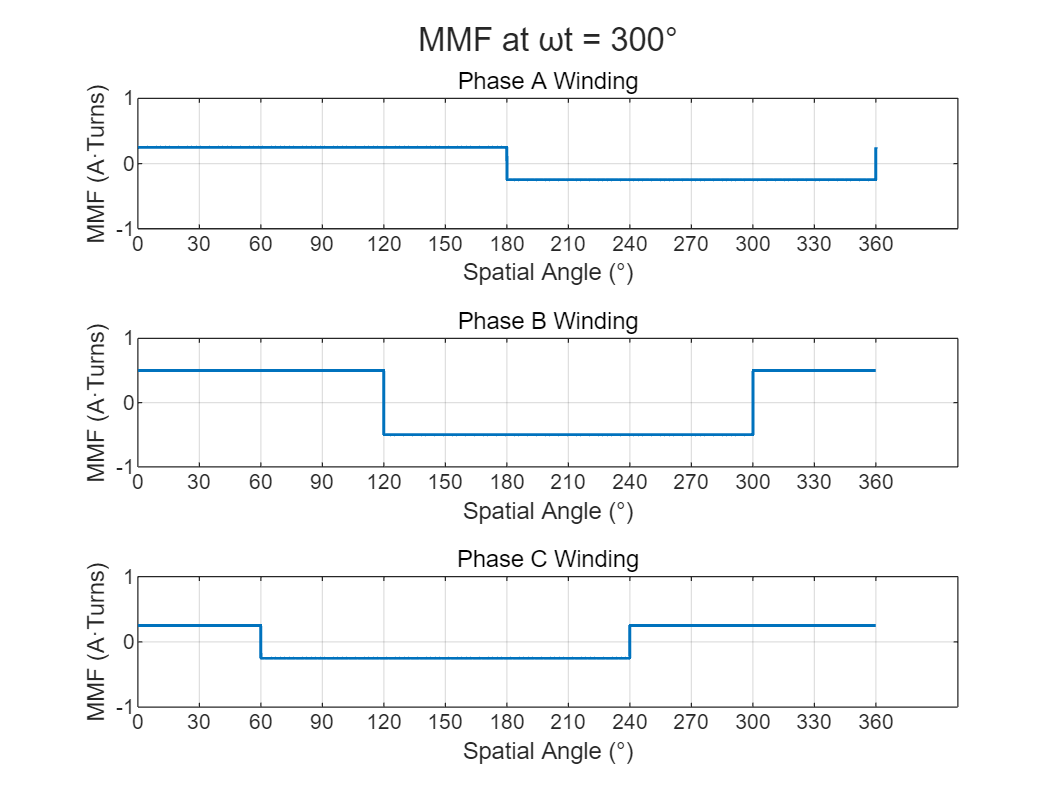
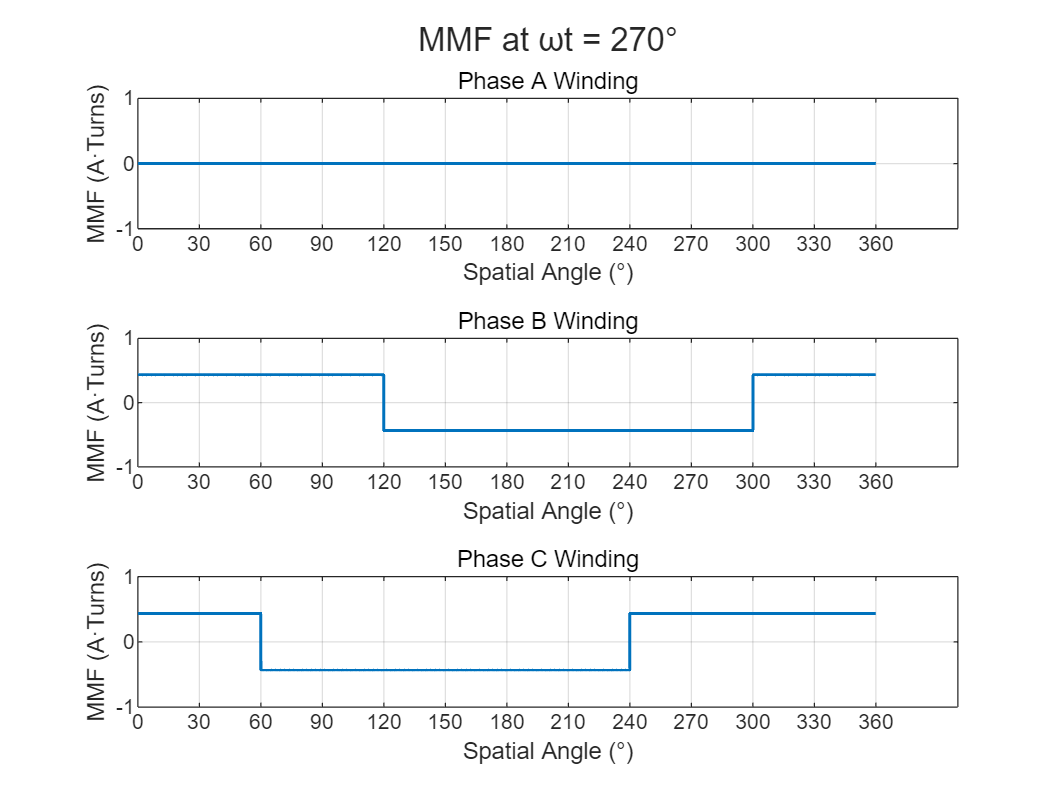
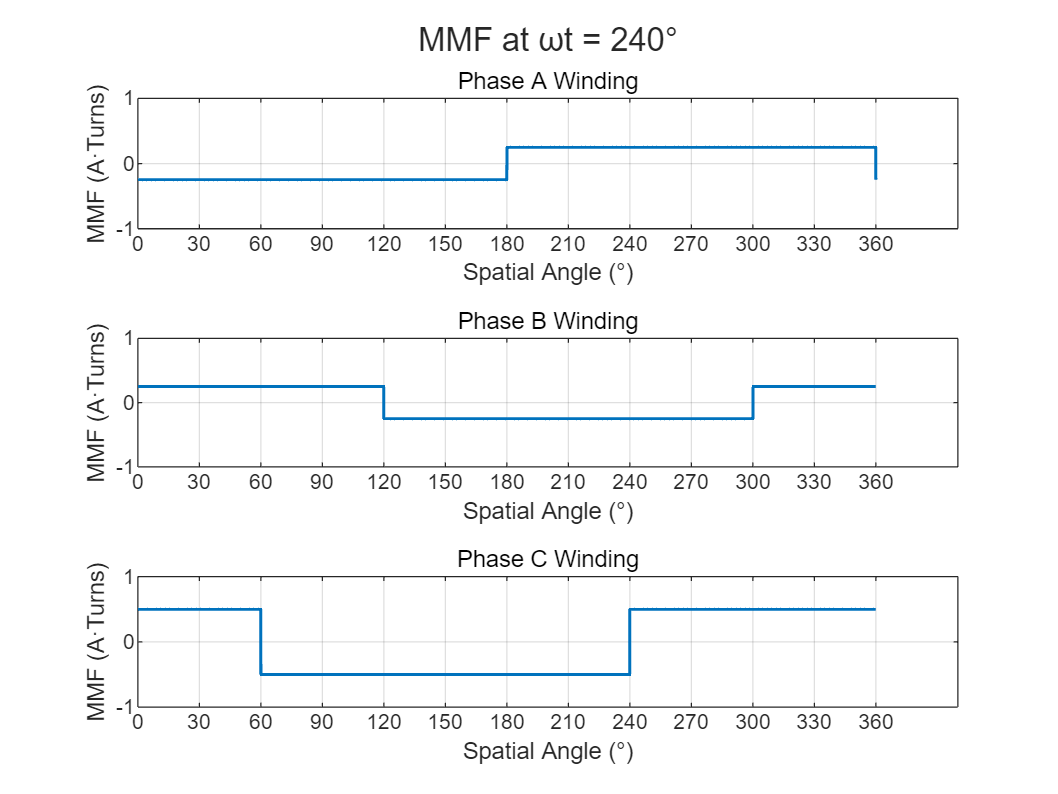
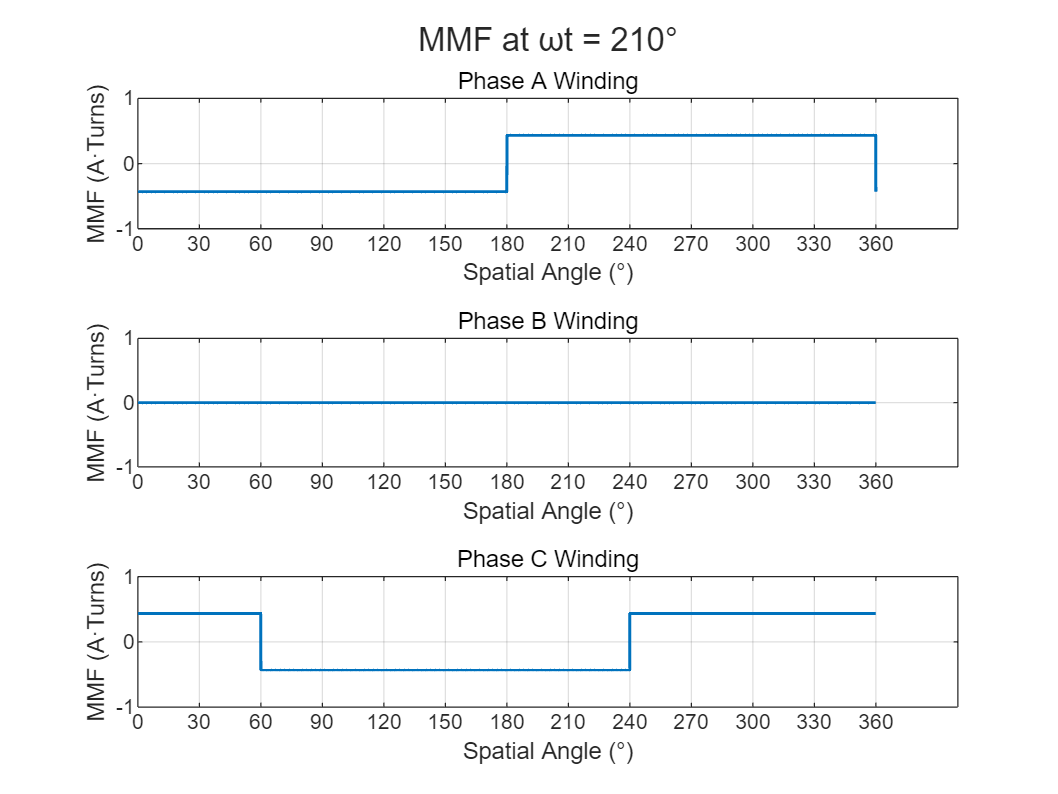
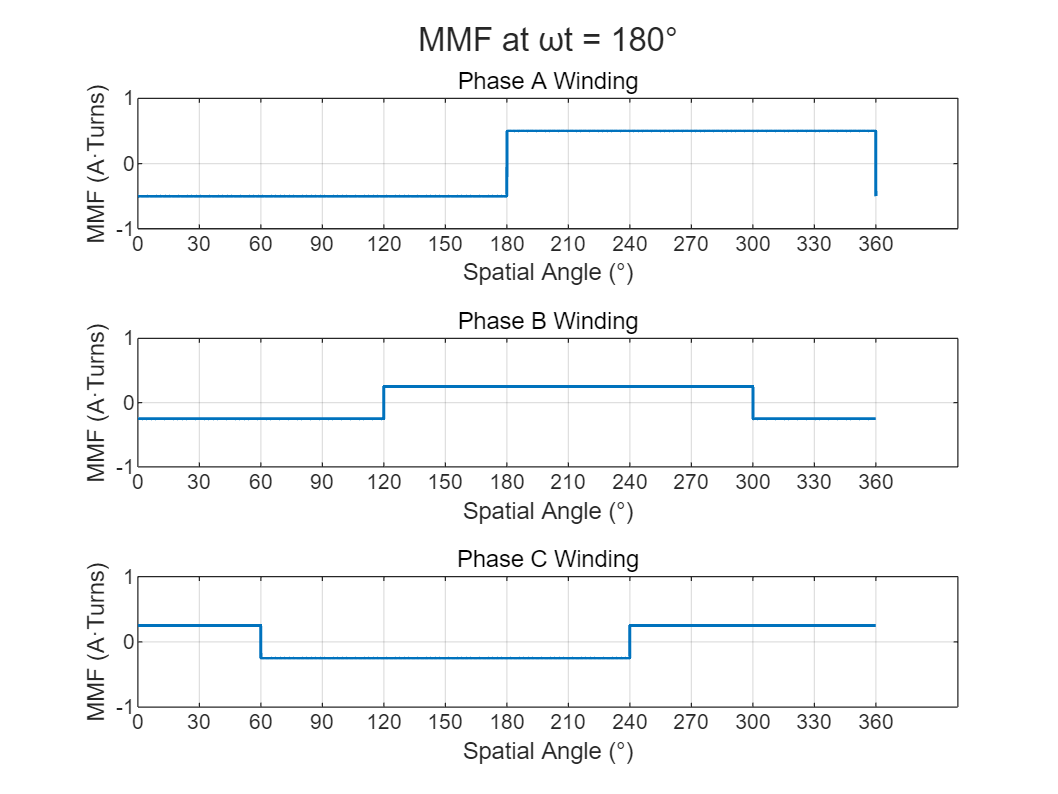
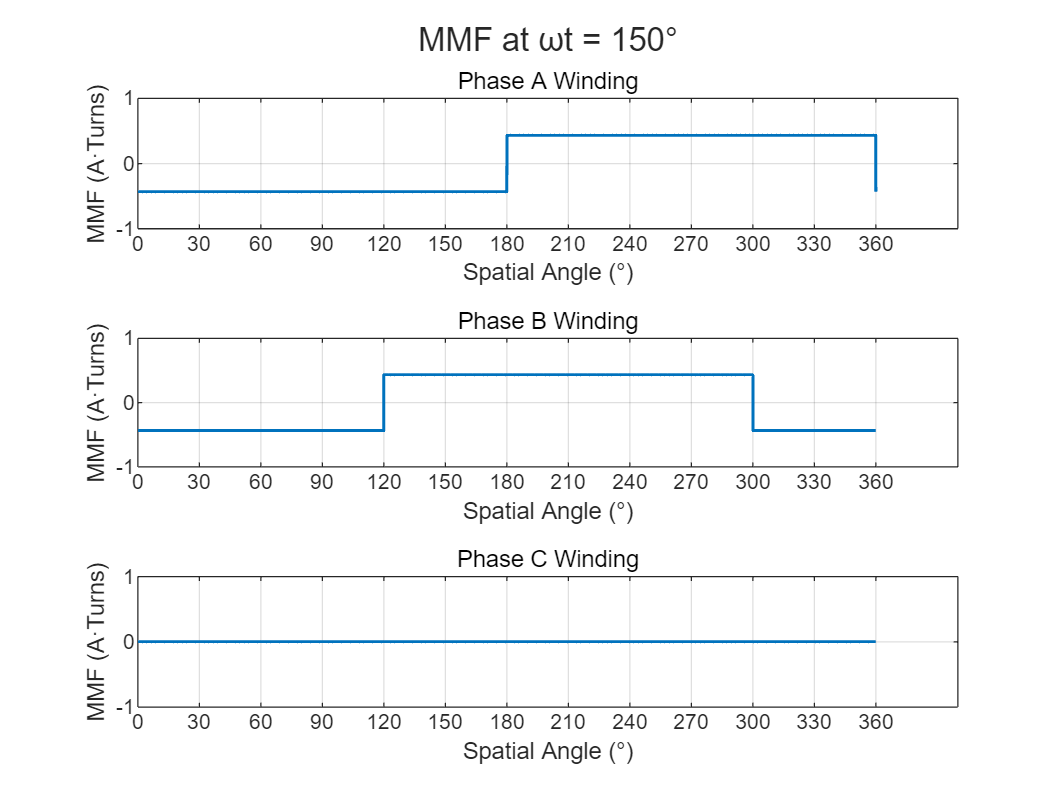
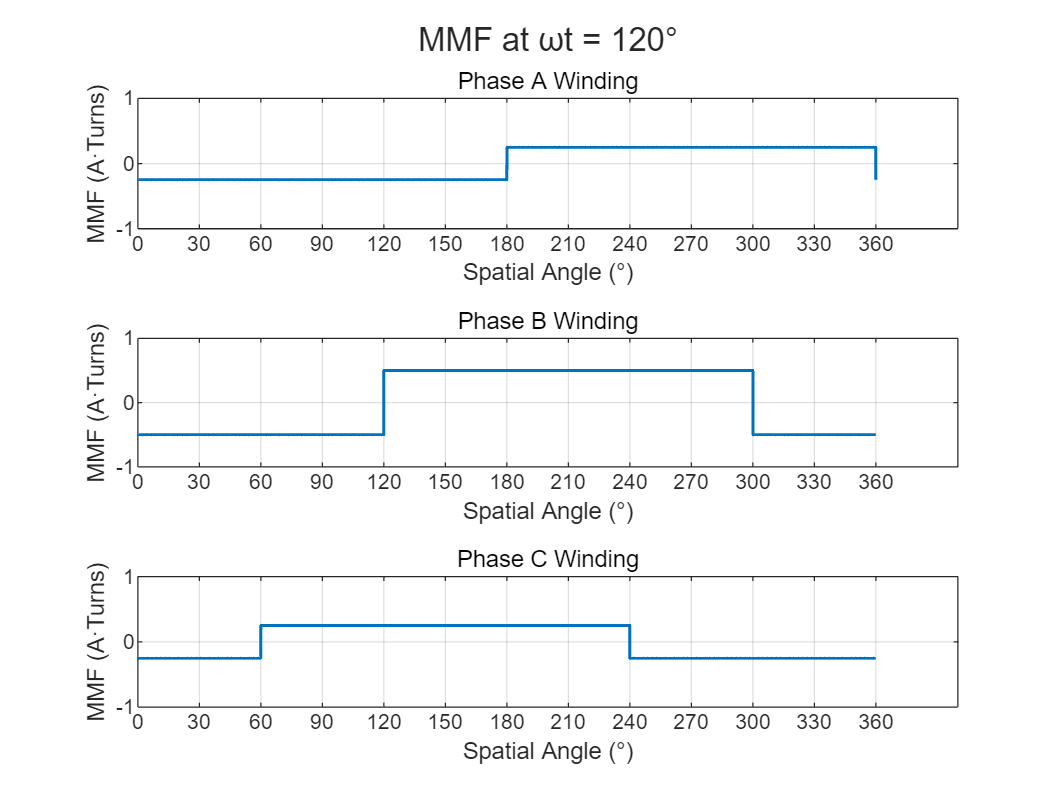
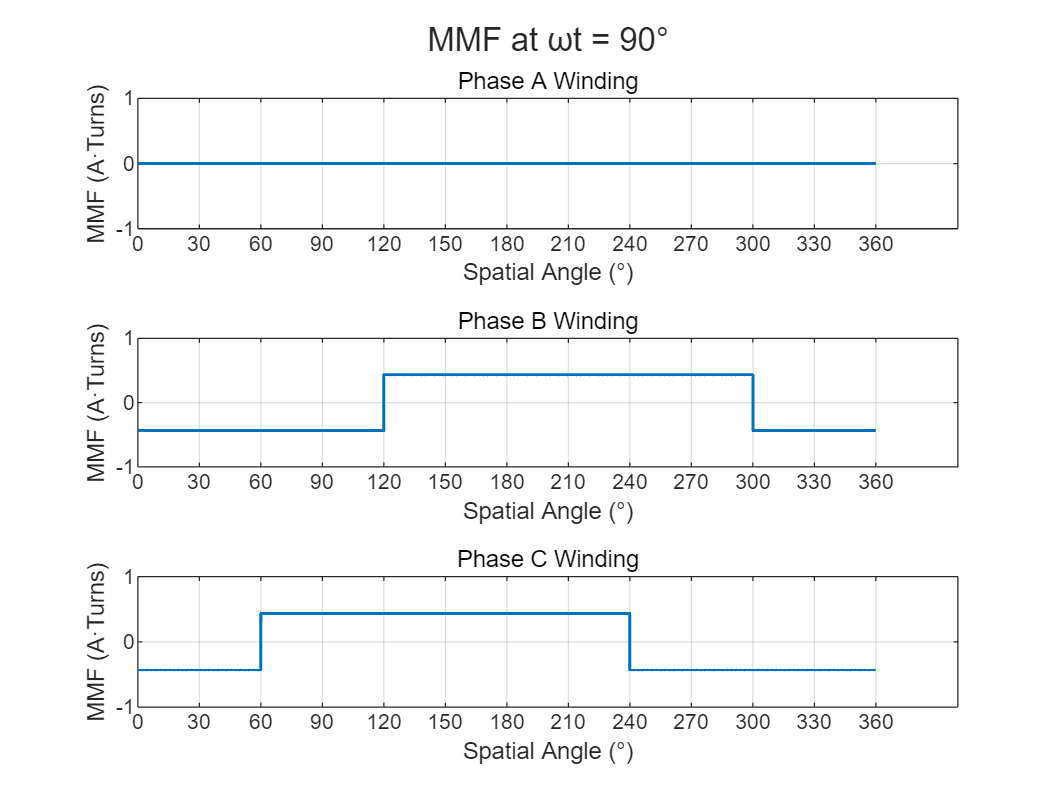
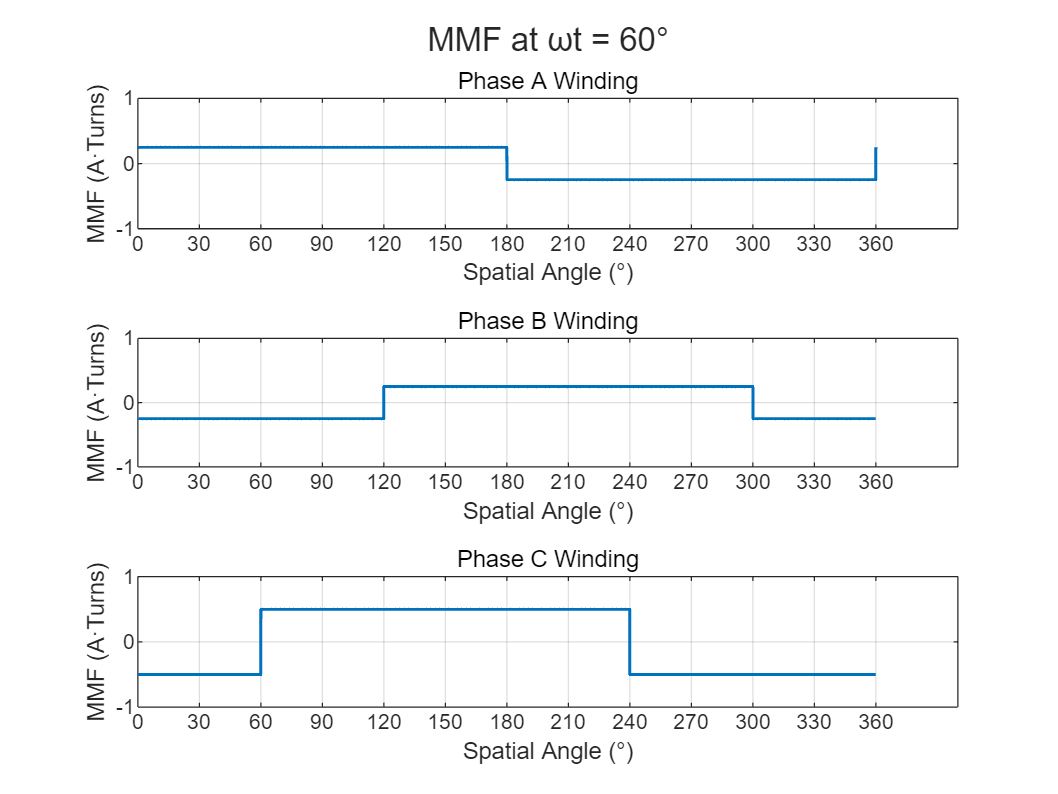
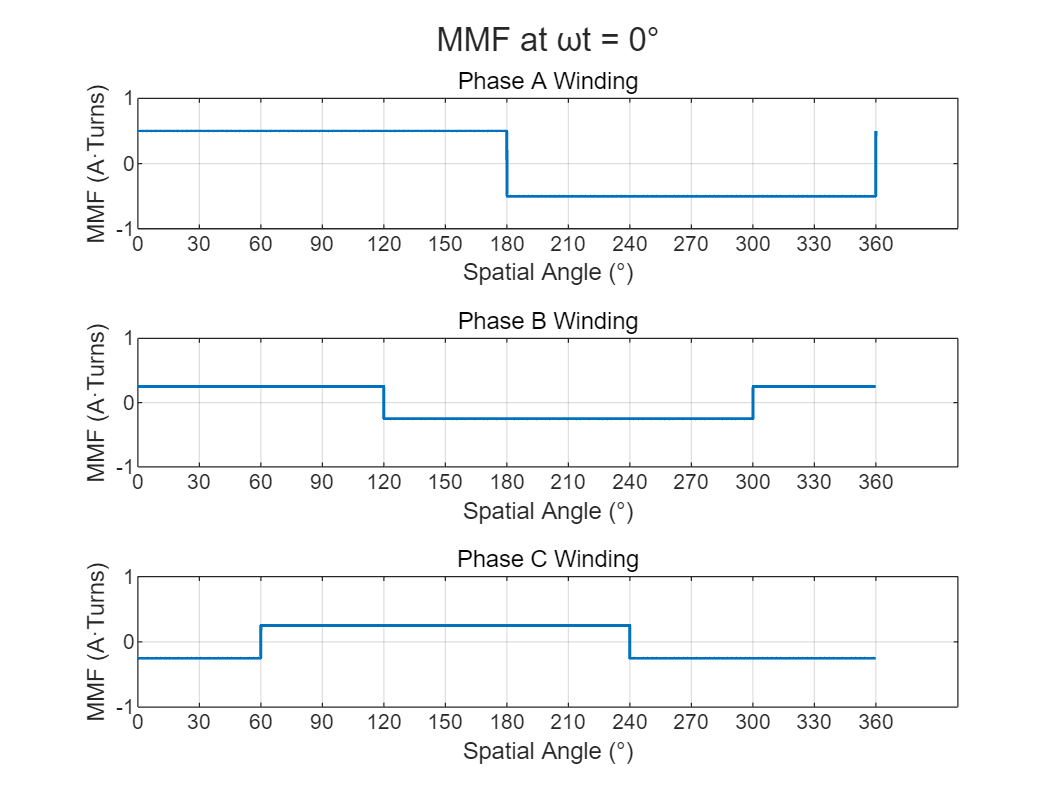
% Plot Phase C MMF

nexttile;

plot(wt\_degrees, MMFC(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase C Winding');

end



function setPlotFormatting(subtitleText)

% Helper function to set formatting for a plot

ylim([-1 1]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("MMF (A·Turns)");

subtitle(subtitleText);

xticks(0:30:360);

end

## (c) Total MMF at Angles of Interest

Find the total MMF at ωt = 0°, 30° , 60° , 90° , 120°, ...

totalMMF = MMFA + MMFB + MMFC;

figure;

tiledlayout(3, 4, 'TileSpacing', 'Compact', 'Padding', 'Compact');

sgtitle('Total MMF at Different ωt Degrees');

titleHandle.FontSize = 13;

for k = 1:length(angleOfInterest)

nexttile;

plot(wt\_degrees, totalMMF(k,:), 'LineWidth', 1.2);

ylim([-2 2]);

yticks(-2:0.5:2)

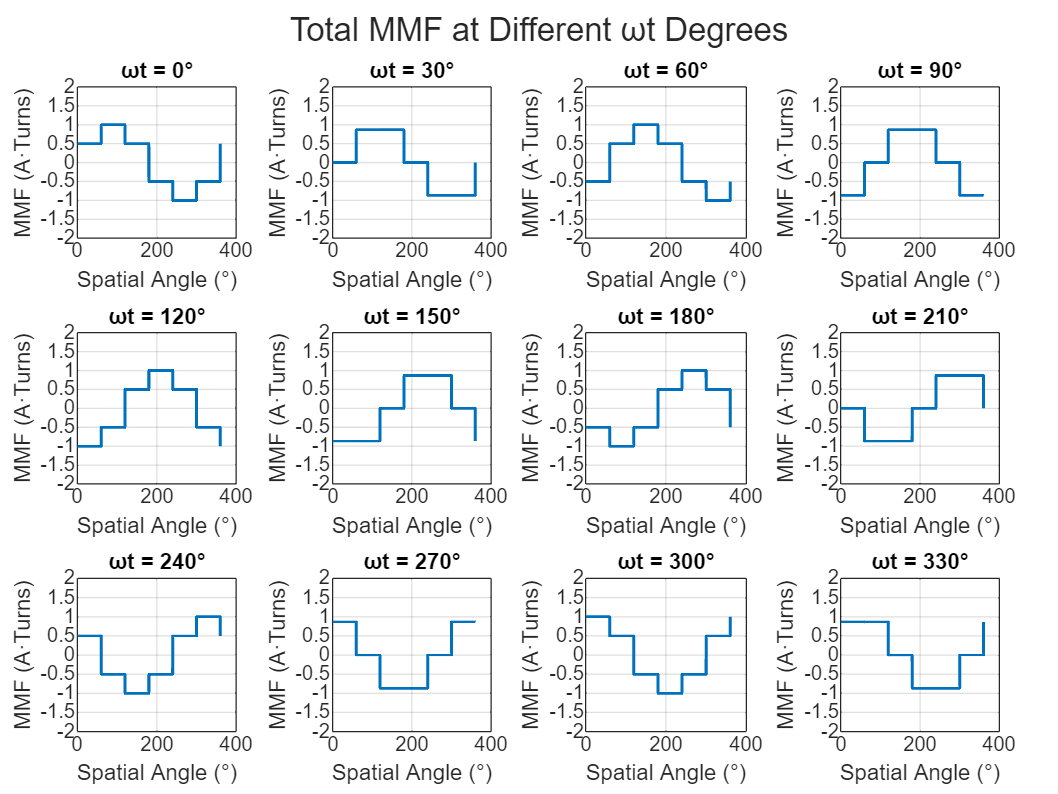
grid on;

xlabel("Spatial Angle (°)");

ylabel("MMF (A·Turns)");

title(sprintf('ωt = %.0f°', angleOfInterest(k)))

end



# Problem 2

## (d) Fourier Series Analysis

### Winding Functions Represented by Fourier Series

% a0 = (1/T)\*int(expression, theta, 0, T);

% an = (2/T)\*int(expression\*cos(n\*theta), theta, 0, T);

% bn = (2/T)\*int(expression\*sin(n\*theta), theta, 0, T);

% DC component

NA\_a0 = mean(NA)

NA\_a0 = -5.1816e-15

% Order of harmonics

orderOfHarmonics = 1:13;

N = 1;

NA\_bn\_ref = 4./(orderOfHarmonics\*pi)\*N/2.\*sin(orderOfHarmonics\*pi/2)

NA\_bn\_ref = *1×13*

0.6366 0.0000 -0.2122 -0.0000 0.1273 0.0000 -0.0909 ⋯

% Initialization

NA\_an = zeros(size(orderOfHarmonics));

NA\_bn = zeros(size(orderOfHarmonics));

for n = 1:length(orderOfHarmonics)

% Amplitude of even harmonics (cos terms)

NA\_an(n) = 2\*mean(NA.\*cos(n\*theta));

% Amplitude of odd harmonics (sin terms)

NA\_bn(n) = 2\*mean(NA.\*sin(n\*theta));

end

NA\_an

NA\_an = *1×13*

10-3 ×

0.3999 0.3999 0.3999 0.3999 0.3999 0.3999 0.3999 ⋯

NA\_bn

NA\_bn = *1×13*

0.6365 -0.0000 0.2122 -0.0000 0.1273 -0.0000 0.0909 ⋯

[NB\_a0, NB\_an, NB\_bn] = computeHarmonicsAmplitude(NB, theta, orderOfHarmonics)

NB\_a0 = -3.7291e-15

NB\_an = *1×13*

-0.5515 -0.0005 -0.0001 -0.0005 0.1100 0.0004 -0.0790 ⋯

NB\_bn = *1×13*

-0.3182 -0.0005 0.2122 0.0005 -0.0637 -0.0000 -0.0454 ⋯

[NC\_a0, NC\_an, NC\_bn] = computeHarmonicsAmplitude(NC, theta, orderOfHarmonics)

NC\_a0 = -3.7291e-15

NC\_an = *1×13*

0.5511 0.0001 0.0009 0.0001 -0.1104 0.0004 0.0786 ⋯

NC\_bn = *1×13*

-0.3189 0.0002 0.2122 -0.0002 -0.0630 -0.0000 -0.0461 ⋯

% Reconstructed winding function

NA\_rec = zeros(size(theta));

NB\_rec = zeros(size(theta));

NC\_rec = zeros(size(theta));

for n = 1:length(orderOfHarmonics)

NA\_rec = NA\_rec + NA\_an(n).\*cos(n\*theta) + NA\_bn(n).\*sin(n\*theta);

if n == orderOfHarmonics(end)

NA\_rec = NA\_rec + NA\_a0/2;

end

end

NB\_rec = reconstructFromFourierSeries(NB\_a0, NB\_an, NB\_bn, theta, orderOfHarmonics);

NC\_rec = reconstructFromFourierSeries(NC\_a0, NC\_an, NC\_bn, theta, orderOfHarmonics);

function result = reconstructFromFourierSeries(a0, an, bn, theta, orderOfHarmonics)

result = zeros(size(theta));

for n = 1:length(orderOfHarmonics)

result = result + an(n).\*cos(n\*theta) + bn(n).\*sin(n\*theta);

if n == orderOfHarmonics(end)

result = result + a0/2;

end

end

end

figure

tiledlayout(3,1)

titleHandle = sgtitle("Winding Functions Reconstructed from Fourier Series");

titleHandle.FontSize = 12;

nexttile;

plot(wt\_degrees,NA\_rec,wt\_degrees,NA,"LineWidth",1.2)

subtitle("Phase A Winding");

ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,NB\_rec,wt\_degrees,NB,"LineWidth",1.2)

subtitle("Phase B Winding");

ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,NC\_rec,wt\_degrees,NC,"LineWidth",1.2)

subtitle("Phase C Winding");

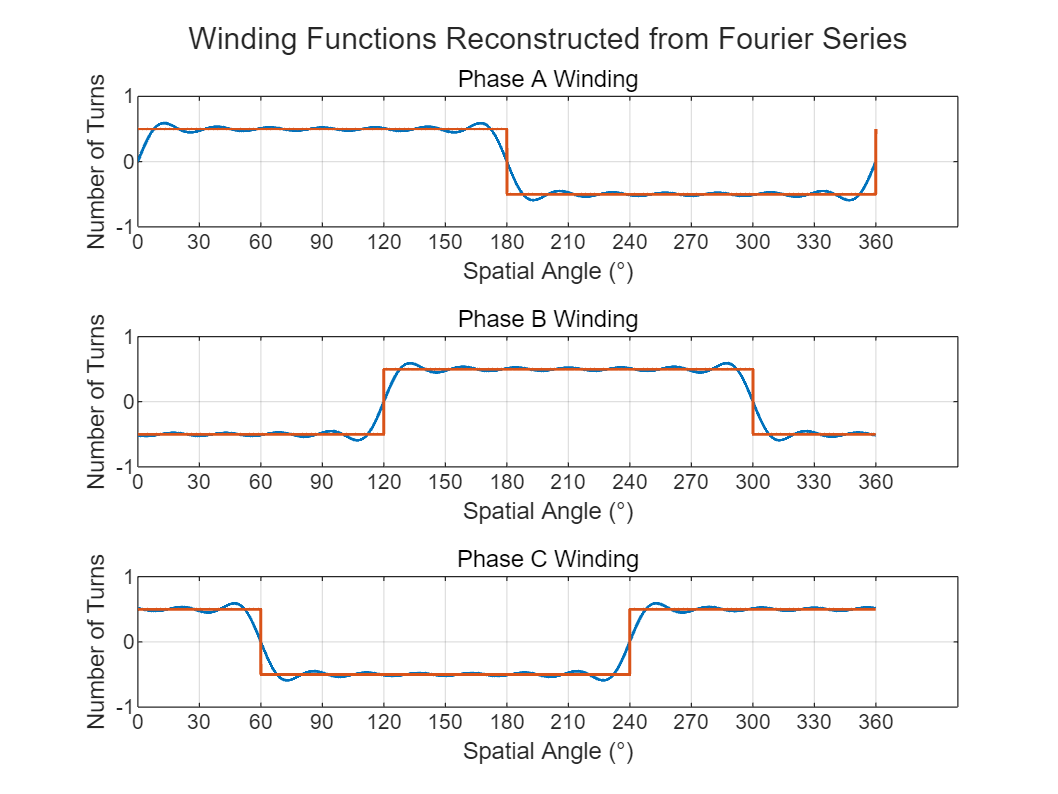
ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid



% Compare with original ones

**Plot Winding Function Odd Harmonics Spectrum**

% numOfPhase = 3;

% N\_bn = [];

% for k = 1:length(orderOfHarmonics)

% N\_bn = [N\_bn; NA\_bn(k) NB\_bn(k) NC\_bn(k)];

% end

%

% figure;

% bar(N\_bn, 'grouped'); % 'grouped' creates the grouped bar chart

%

% % Labeling

% xlabel('Order of Harmonics');

% ylabel('Magnitude');

% title('Odd Harmonics Spectrum of Phase A, B, C Winding Functions');

%

% % Customize the x-tick labels

% % set(gca, 'XTickLabel', {'Group 1', 'Group 2', 'Group 3'});

%

% legend({'Winding A', 'Winding B', 'Winding C'}, 'Location', 'best');

% grid on;

### Sinusoidal Currents Represented by Fourier Series

[iA\_a0, iA\_an, iA\_bn] = computeHarmonicsAmplitude(iA, wt, orderOfHarmonics)

iA\_a0 = 2.0000e-04

iA\_an = *1×13*

1.0002 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 ⋯

iA\_bn = *1×13*

10-16 ×

0.2467 0.1715 0.3035 0.2122 -0.1285 -0.3443 -0.1322 ⋯

[iB\_a0, iB\_an, iB\_bn] = computeHarmonicsAmplitude(iB, wt, orderOfHarmonics)

iB\_a0 = -1.0000e-04

iB\_an = *1×13*

-0.5001 -0.0002 -0.0002 -0.0002 -0.0002 -0.0002 -0.0002 ⋯

iB\_bn = *1×13*

0.8659 -0.0000 0.0000 0.0000 0.0000 -0.0000 0.0000 ⋯

[iC\_a0, iC\_an, iC\_bn] = computeHarmonicsAmplitude(iC, wt, orderOfHarmonics)

iC\_a0 = -1.0000e-04

iC\_an = *1×13*

-0.5001 -0.0002 -0.0002 -0.0002 -0.0002 -0.0002 -0.0002 ⋯

iC\_bn = *1×13*

-0.8659 -0.0000 0.0000 0.0000 0.0000 0.0000 -0.0000 ⋯

iA\_rec = reconstructFromFourierSeries(iA\_a0, iA\_an, iA\_bn, wt, orderOfHarmonics);

iB\_rec = reconstructFromFourierSeries(iB\_a0, iB\_an, iB\_bn, wt, orderOfHarmonics);

iC\_rec = reconstructFromFourierSeries(iC\_a0, iC\_an, iC\_bn, wt, orderOfHarmonics);

figure

tiledlayout(3,1)

titleHandle = sgtitle("Phase Currents Reconstructed from Fourier Series");

titleHandle.FontSize = 12;

nexttile;

plot(wt\_degrees,iA\_rec, ...

"DisplayName", "Reconstructed", ...

"LineWidth",1.2, ...

"Marker","o")

hold on

plot(wt\_degrees,iA, ...

"DisplayName", "Original", ...

"LineWidth",1.2)

hold off

legend

subtitle("Phase A Current");

ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,iB\_rec,wt\_degrees,iB,"LineWidth",1.2)

subtitle("Phase B Current");

ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,iC\_rec,wt\_degrees,iC,"LineWidth",1.2)

subtitle("Phase C Current");

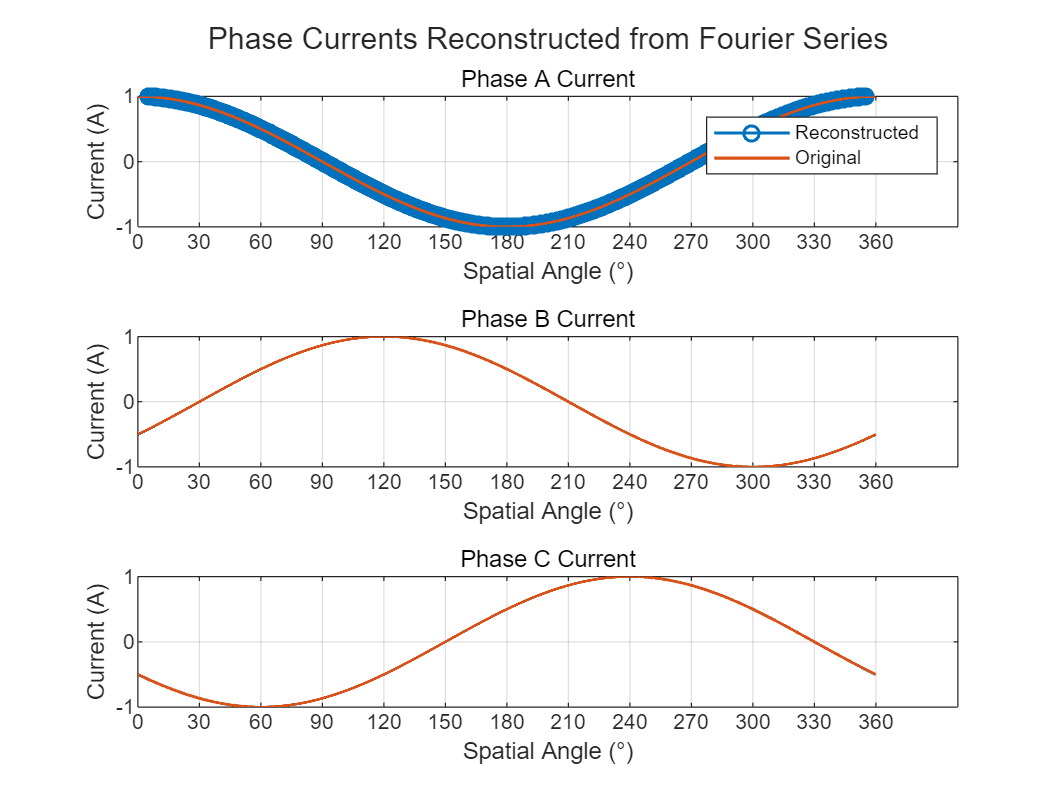
ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid



### Phase MMF Represented by Fourier Series at Angles of Interest

iA\_rec\_angleOfInterest = iA\_rec(indices);

iB\_rec\_angleOfInterest = iB\_rec(indices);

iC\_rec\_angleOfInterest = iC\_rec(indices);

MMFA\_rec = NA\_rec.\*iA\_rec\_angleOfInterest'; % Transposed

MMFB\_rec = NB\_rec.\*iB\_rec\_angleOfInterest';

MMFC\_rec = NC\_rec.\*iC\_rec\_angleOfInterest';

### Total MMF Represented by Fourier Series at Angles of Interest

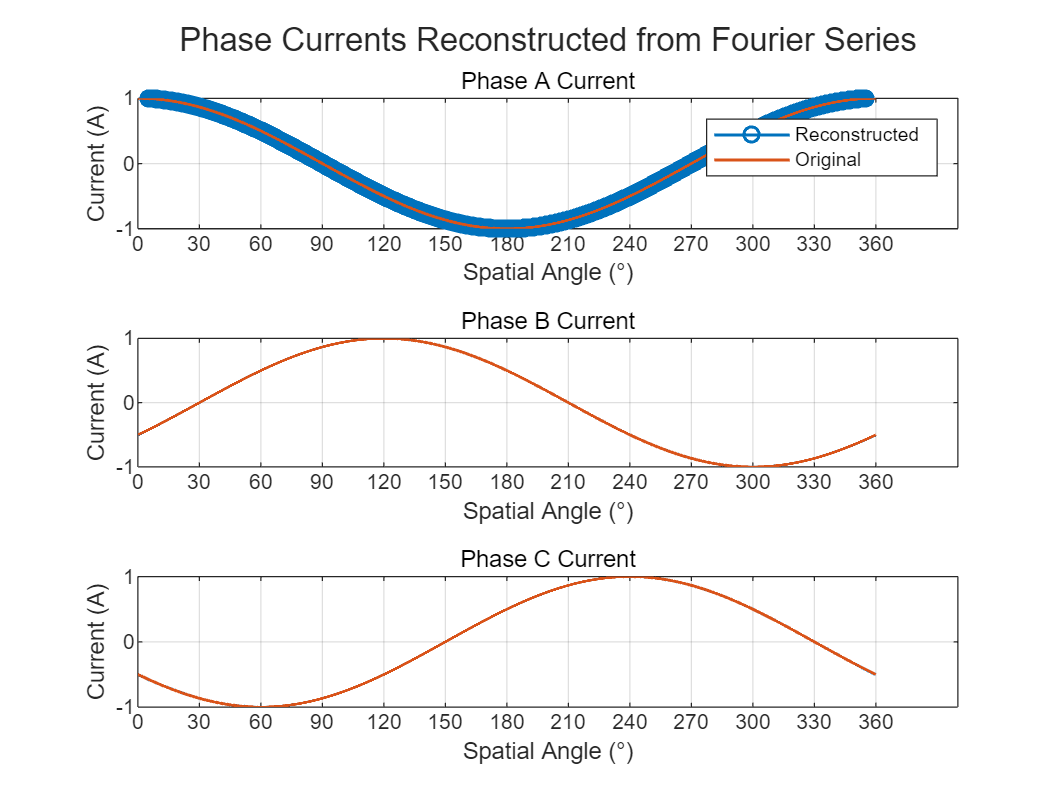
totalMMF\_rec = MMFA\_rec + MMFB\_rec + MMFC\_rec;

figure;

tiledlayout(3, 4, 'TileSpacing', 'Compact', 'Padding', 'Compact');

sgtitle('Total MMF at Different ωt Degrees');

titleHandle.FontSize = 13;



for k = 1:length(angleOfInterest)

nexttile;

plot(wt\_degrees, totalMMF\_rec(k,:), 'LineWidth', 1.2);

hold on

plot(wt\_degrees, totalMMF(k,:), 'LineWidth', 0.8);

hold off

ylim([-2 2]);

yticks(-2:0.5:2)

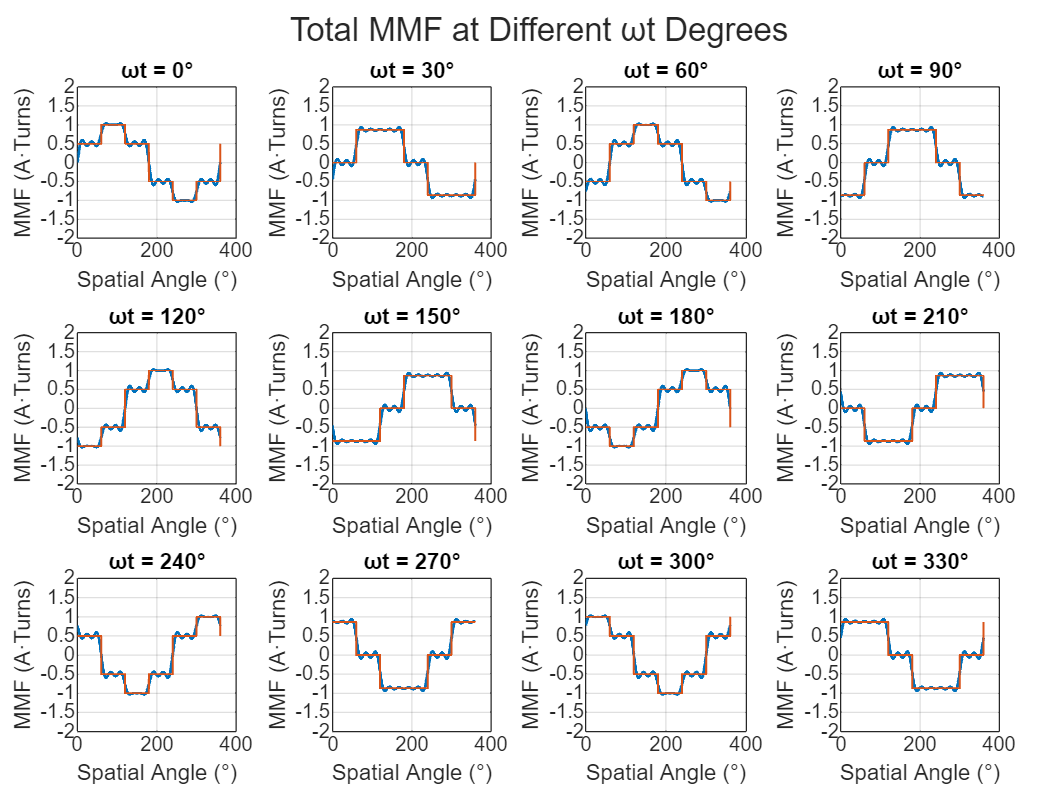
grid on;

xlabel("Spatial Angle (°)");

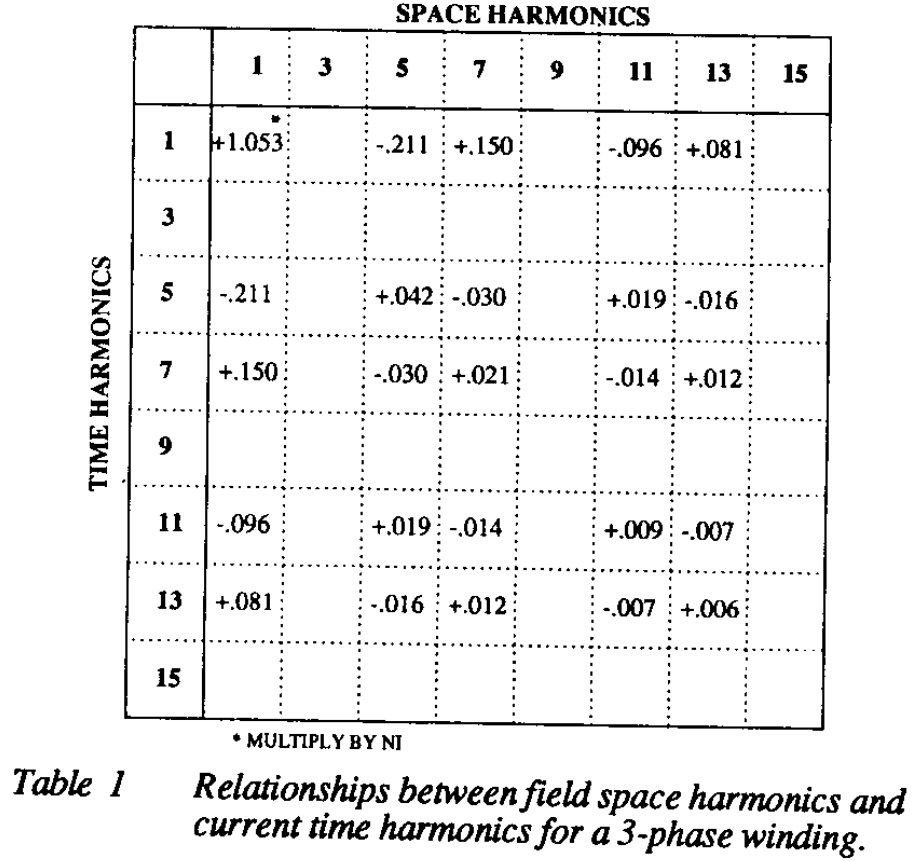
ylabel("MMF (A·Turns)");

title(sprintf('ωt = %.0f°', angleOfInterest(k)))

end



### Table



**Space Harmonics** (on the top horizontal axis): These refer to the harmonic components of the winding functions. The harmonics listed here include 1st, 2nd, 3rd...all the way to 13th order harmonics.

**Time Harmonics** (on the left vertical axis): These correspond to the harmonic components of the current in the winding. The harmonics listed here include 1st, 2nd, 3rd...all the way to 13th order harmonics.

harmonic\_matrix = zeros(orderOfHarmonics(end));

diagnal\_harmonics = diag(NA\_bn.\*iA\_bn + NB\_bn.\*iB\_bn + NC\_an.\*iC\_bn);

cross\_harmonics = NA\_bn.\*iA\_bn' + NB\_bn.\*iB\_bn' + NC\_an.\*iC\_bn';

harmonic\_matrix(harmonic\_matrix==0) = cross\_harmonics(harmonic\_matrix==0)

harmonic\_matrix = *13×13*

-0.7526 -0.0005 0.1829 0.0004 0.0404 -0.0003 -0.1073 ⋯

-0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 -0.0000

0.0000 -0.0000 0.0000 0.0000 -0.0000 0.0000 0.0000

0.0000 -0.0000 0.0000 0.0000 -0.0000 0.0000 0.0000

-0.0000 -0.0000 0.0000 0.0000 -0.0000 0.0000 -0.0000

0.0000 0.0000 -0.0000 -0.0000 -0.0000 0.0000 0.0000

-0.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000

-0.0000 -0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000

0.0000 0.0000 -0.0000 -0.0000 0.0000 -0.0000 0.0000

0.0000 0.0000 -0.0000 -0.0000 -0.0000 0.0000 0.0000

⋮

% I'm confident with this method

function [a0,an,bn] = computeHarmonicsAmplitude(expr,wt,orderOfHarmonics)

a0 = mean(expr);

% Initialization

an = zeros(size(orderOfHarmonics));

bn = zeros(size(orderOfHarmonics));

for n = 1:length(orderOfHarmonics)

% Amplitude of even harmonics (cos terms)

an(n) = 2\*mean(expr.\*cos(n\*wt));

% Amplitude of odd harmonics (sin terms)

bn(n) = 2\*mean(expr.\*sin(n\*wt));

end

end

ECEN 611 Homework 2: Six Step

Shuxuan Chen | UIN: 132006082 | Fall 2024

*Last Updated: Sep 22*

Table of Contents

[Problem 1](#MW_H_4A3FD973)   
 [(a) Winding Function Diagrams for Windings A, B, and C](#MW_H_B9DB3A63)   
 [General Settings](#MW_H_8C99DF8F)   
 [Counting Functions for Windings A, B, and C](#MW_H_C9AFDCE8)   
 [Winding Functions for Windings A, B, and C](#MW_H_4AB1FE44)   
 [Plot Counting Functions and Winding Functions A, B, and C](#MW_H_564D14E3)   
 [(b) MMF of Windings A, B, and C Supplied by Three Phase Six Step Currents](#MW_H_5C3769CE)   
 [Three Phase Six Step Currents](#MW_H_3742C372)   
 [Three Phase Currents at Angle of Interest](#MW_H_4782FEB6)   
 [Three Phase MMFs at Angles of Interest](#MW_H_E880D25D)   
 [(c) Total MMF at Angles of Interest](#MW_H_F4BE4604)   
[Problem 2](#MW_H_65280B52)   
 [(d) Fourier Series Analysis](#MW_H_4A9044CF)   
 [Winding Functions Represented by Fourier Series](#MW_H_77276C02)   
 [Six Step Currents Represented by Fourier Series](#MW_H_B9AF31C1)   
 [Phase MMF Represented by Fourier Series at Angles of Interest](#MW_H_D74B67B5)   
 [Total MMF Represented by Fourier Series at Angles of Interest](#MW_H_5F8DFCEB)   
 [Table](#MW_H_0CFFF6FC)

A diagram of a circle with arrows and circles

Description automatically generated

clearvars

clc

Numeric computation is always recommended when it can solve the problem with acceptable precision.

# Problem 1

## (a) Winding Function Diagrams for Windings A, B, and C

Draw the winding functions for windings A, B, and C on the same sheet of paper.

### General Settings

% Define the spatial angle from 0 to 360 degrees

T = 2\*pi;

T\_degree = rad2deg(T)

T\_degree = 360

numOfPoints = 5000; % This affects precision of Fourier analysis

theta = linspace(0, T, numOfPoints);

theta\_degrees = linspace(0, T\_degree, numOfPoints)

theta\_degrees = *1×5000*

0 0.0720 0.1440 0.2160 0.2881 0.3601 0.4321 ⋯

### Counting Functions for Windings A, B, and C

% Initialize counting function to zero arrays

nA = zeros(size(theta));

nB = zeros(size(theta));

nC = zeros(size(theta));

% Define nA

nA(mod(theta, T) >= 0 & mod(theta, T) < T/2) = 1;

nA(mod(theta, T) >= T/2 & mod(theta, T) <= T) = 0;

**Explanation:**

* **mod(theta, T)**: This ensures that the function repeats every T. By applying mod(theta, T), you're effectively folding the waveform to repeat from T to 2T.
* **First Condition**: For the interval 0≤θ≤T/2, the function returns 1.
* **Second Condition**: For the interval T/2<θ≤T, the function returns nA\_pi - 1.

% Define nB

% nB(theta >= 0 & theta < 120) = 0;

% nB(theta >= 120 & theta < 300) = 1;

% nB(theta >= 300 & theta <= 360) = 0;

%

% % Define nC

% nC(theta >= 0 & theta < 60) = 1;

% nC(theta >= 60 & theta < 240) = 0;

% nC(theta >= 240 & theta <= 360) = 1;

% Define phase shift for iB and iC

phase\_shift = 2\*pi/3;

% Define iB as phase-shifted iA

nB = circshift(nA, [0, round(phase\_shift/(2\*pi) \* length(theta))]);

% Define iC as phase-shifted iA by 2\*phase\_shift

nC = circshift(nA, [0, round(2\*phase\_shift/(2\*pi) \* length(theta))]);

### Winding Functions for Windings A, B, and C

% Define NA, NB, and NC

NA = nA - mean(nA);

NB = nB - mean(nB);

NC = nC - mean(nC);

### Plot Counting Functions and Winding Functions A, B, and C

figure

tiledlayout(3,1)

titleHandle = sgtitle("Counting Function and Winding Function");

titleHandle.FontSize = 16;

% First tile: nA and NA

nexttile;

plot(theta\_degrees, nA, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NA, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase A Winding");

legend('nA', 'NA');

xticks(0:30:360);

% Second tile: nB and NB

nexttile;

plot(theta\_degrees, nB, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NB, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase B Winding");

legend('nB', 'NB');

xticks(0:30:360);

% Third tile: nC and NC

nexttile;

plot(theta\_degrees, nC, 'LineWidth', 1.2);

hold on;

plot(theta\_degrees, NC, 'LineWidth', 1.2);

hold off;

ylim([-1 2]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("Number of Turns");

subtitle("Phase C Winding");

legend('nC', 'NC');

xticks(0:30:360);

A graph of a function

Description automatically generated with medium confidence

% Counting functions √

% Winding functions √

## (b) MMF of Windings A, B, and C Supplied by Three Phase Six Step Currents

Draw the MMF of each phase if the windings are supplied by three phase Six Step currents at ωt = 0°, 30° , 60° , 90° , 120°, ...

### Three Phase Six Step Currents

A table of mathematical equations

Description automatically generated

wt = linspace(0, T, numOfPoints);

wt\_degrees = rad2deg(wt);

iA = zeros(size(wt));

iB = zeros(size(wt));

iC = zeros(size(wt));

% Define iA (piecewise function for phase A)

iA(mod(wt, 2\*pi) >= 0 & mod(wt, 2\*pi) < pi/3) = 1/2;

iA(mod(wt, 2\*pi) >= pi/3 & mod(wt, 2\*pi) < 2\*pi/3) = -1/2;

iA(mod(wt, 2\*pi) >= 2\*pi/3 & mod(wt, 2\*pi) < pi) = -1;

iA(mod(wt, 2\*pi) >= pi & mod(wt, 2\*pi) < 4\*pi/3) = -1/2;

iA(mod(wt, 2\*pi) >= 4\*pi/3 & mod(wt, 2\*pi) < 5\*pi/3) = 1/2;

iA(mod(wt, 2\*pi) >= 5\*pi/3 & mod(wt, 2\*pi) <= 2\*pi) = 1;

% Define phase shift for iB and iC

phase\_shift = 2\*pi/3;

% Define iB as phase-shifted iA

iB = circshift(iA, [0, round(phase\_shift/(2\*pi) \* length(wt))]);

% Define iC as phase-shifted iA by 2\*phase\_shift

iC = circshift(iA, [0, round(2\*phase\_shift/(2\*pi) \* length(wt))]);

% Ensure wt\_degrees, iA, iB, and iC are defined

figure;

plot(wt\_degrees, iA, ...

wt\_degrees, iB, ...

wt\_degrees, iC, 'LineWidth', 1.2);

legend('iA', 'iB', 'iC');

title('Three-Phase Six Step Currents');

xlabel('Degrees');

ylabel('Current (A)');

ylim([-2 2])

grid on;

A graph with different colored lines

Description automatically generated

### Three Phase Currents at Angle of Interest

angleOfInterest = 0:30:330;

% Pre-allocate the index array

indices = zeros(size(angleOfInterest));

% Loop through each angle of interest and find the closest match in wt\_degrees

for i = 1:length(angleOfInterest)

[~, idx] = min(abs(wt\_degrees - angleOfInterest(i))); % Find closest value

indices(i) = idx; % Store the index of the closest value

end

iA\_angleOfInterest = iA(indices);

iB\_angleOfInterest = iB(indices);

iC\_angleOfInterest = iC(indices);

% Ensure wt\_degrees, iA, iB, and iC are defined

figure;

plot(wt\_degrees(indices), iA\_angleOfInterest, 'o', ...

wt\_degrees(indices), iB\_angleOfInterest, 'o', ...

wt\_degrees(indices), iC\_angleOfInterest, 'o', ...

'LineWidth', 1.2);

legend('iA', 'iB', 'iC');

title('Three-Phase Six Step Currents (at Angle of Interest)');

xlabel('Degrees');

ylabel('Current (A)');

ylim([-2 2])

grid on;

A graph of a number of degrees

Description automatically generated with medium confidence

### Three Phase MMFs at Angles of Interest

MMFA = NA .\* iA\_angleOfInterest';

MMFB = NB .\* iB\_angleOfInterest';

MMFC = NC .\* iC\_angleOfInterest';

for k = 1:length(angleOfInterest)

figure;

tiledlayout(3, 1);

% Common title with current wt in degrees

titleHandle = sgtitle(sprintf('MMF at ωt = %.0f°', angleOfInterest(k)));

titleHandle.FontSize = 13;

% Plot Phase A MMF

nexttile;

plot(wt\_degrees, MMFA(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase A Winding');

% Plot Phase B MMF

nexttile;

plot(wt\_degrees, MMFB(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase B Winding');

% Plot Phase C MMF

nexttile;

plot(wt\_degrees, MMFC(k,:), 'LineWidth', 1.2);

setPlotFormatting('Phase C Winding');

end

A diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a diagram

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidenceA diagram of a phase and a wind turbine

Description automatically generated with medium confidence

function setPlotFormatting(subtitleText)

% Helper function to set formatting for a plot

ylim([-1 1]);

grid on;

xlabel("Spatial Angle (°)");

ylabel("MMF (A·Turns)");

subtitle(subtitleText);

xticks(0:30:360);

end

## (c) Total MMF at Angles of Interest

Find the total MMF at ωt = 0°, 30° , 60° , 90° , 120°, ...

totalMMF = MMFA + MMFB + MMFC;

figure;

tiledlayout(3, 4, 'TileSpacing', 'Compact', 'Padding', 'Compact');

sgtitle('Total MMF at Different ωt Degrees');

titleHandle.FontSize = 13;

for k = 1:length(angleOfInterest)

nexttile;

plot(wt\_degrees, totalMMF(k,:), 'LineWidth', 1.2);

ylim([-2 2]);

yticks(-2:0.5:2)

grid on;

xlabel("Spatial Angle (°)");

ylabel("MMF (A·Turns)");

title(sprintf('ωt = %.0f°', angleOfInterest(k)))

end

A graph of different angles

Description automatically generated

# Problem 2

## (d) Fourier Series Analysis

### Winding Functions Represented by Fourier Series

% a0 = (1/T)\*int(expression, theta, 0, T);

% an = (2/T)\*int(expression\*cos(n\*theta), theta, 0, T);

% bn = (2/T)\*int(expression\*sin(n\*theta), theta, 0, T);

% DC component

NA\_a0 = mean(NA)

NA\_a0 = -5.1816e-15

% Order of harmonics

orderOfHarmonics = 1:13;

N = 1;

NA\_bn\_ref = 4./(orderOfHarmonics\*pi)\*N/2.\*sin(orderOfHarmonics\*pi/2)

NA\_bn\_ref = *1×13*

0.6366 0.0000 -0.2122 -0.0000 0.1273 0.0000 -0.0909 ⋯

% Initialization

NA\_an = zeros(size(orderOfHarmonics));

NA\_bn = zeros(size(orderOfHarmonics));

for n = 1:length(orderOfHarmonics)

% Amplitude of even harmonics (cos terms)

NA\_an(n) = 2\*mean(NA.\*cos(n\*theta));

% Amplitude of odd harmonics (sin terms)

NA\_bn(n) = 2\*mean(NA.\*sin(n\*theta));

end

% NA\_an

% NA\_bn

[NB\_a0, NB\_an, NB\_bn] = computeHarmonicsAmplitude(NB, theta, orderOfHarmonics)

NB\_a0 = -3.7291e-15

NB\_an = *1×13*

-0.5515 -0.0005 -0.0001 -0.0005 0.1100 0.0004 -0.0790 ⋯

NB\_bn = *1×13*

-0.3182 -0.0005 0.2122 0.0005 -0.0637 -0.0000 -0.0454 ⋯

[NC\_a0, NC\_an, NC\_bn] = computeHarmonicsAmplitude(NC, theta, orderOfHarmonics)

NC\_a0 = -3.7291e-15

NC\_an = *1×13*

0.5511 0.0001 0.0009 0.0001 -0.1104 0.0004 0.0786 ⋯

NC\_bn = *1×13*

-0.3189 0.0002 0.2122 -0.0002 -0.0630 -0.0000 -0.0461 ⋯

% Reconstructed winding function

NA\_rec = zeros(size(theta));

NB\_rec = zeros(size(theta));

NC\_rec = zeros(size(theta));

for n = 1:length(orderOfHarmonics)

NA\_rec = NA\_rec + NA\_an(n).\*cos(n\*theta) + NA\_bn(n).\*sin(n\*theta);

if n == orderOfHarmonics(end)

NA\_rec = NA\_rec + NA\_a0/2;

end

end

NB\_rec = reconstructFromFourierSeries(NB\_a0, NB\_an, NB\_bn, theta, orderOfHarmonics);

NC\_rec = reconstructFromFourierSeries(NC\_a0, NC\_an, NC\_bn, theta, orderOfHarmonics);

function result = reconstructFromFourierSeries(a0, an, bn, theta, orderOfHarmonics)

result = zeros(size(theta));

for n = 1:length(orderOfHarmonics)

result = result + an(n).\*cos(n\*theta) + bn(n).\*sin(n\*theta);

if n == orderOfHarmonics(end)

result = result + a0/2;

end

end

end

figure

tiledlayout(3,1)

titleHandle = sgtitle("Winding Functions Reconstructed from Fourier Series");

titleHandle.FontSize = 12;

nexttile;

plot(wt\_degrees,NA\_rec,wt\_degrees,NA,"LineWidth",1.2)

subtitle("Phase A Winding");

ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,NB\_rec,wt\_degrees,NB,"LineWidth",1.2)

subtitle("Phase B Winding");

ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,NC\_rec,wt\_degrees,NC,"LineWidth",1.2)

subtitle("Phase C Winding");

ylabel("Number of Turns");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

A diagram of a wind turbine

Description automatically generated with medium confidence

% Compare with original ones

**Plot Winding Function Odd Harmonics Spectrum**

% numOfPhase = 3;

N\_bn = [];

for k = 1:length(orderOfHarmonics)

N\_bn = [N\_bn; NA\_bn(k) NB\_bn(k) NC\_bn(k)];

end

figure;

bar(N\_bn, 'grouped'); % 'grouped' creates the grouped bar chart

% Labeling

xlabel('Order of Harmonics');

ylabel('Magnitude');

title('Odd Harmonics Spectrum of Phase A, B, C Winding Functions');

% Customize the x-tick labels

% set(gca, 'XTickLabel', {'Group 1', 'Group 2', 'Group 3'});

legend({'Winding A', 'Winding B', 'Winding C'}, 'Location', 'best');

grid on;

A graph of a diagram

Description automatically generated with medium confidence

### Six Step Currents Represented by Fourier Series

[iA\_a0, iA\_an, iA\_bn] = computeHarmonicsAmplitude(iA, wt, orderOfHarmonics)

iA\_a0 = 2.0000e-04

iA\_an = *1×13*

0.8270 -0.0001 -0.0000 -0.0000 -0.1652 0.0004 0.1183 ⋯

iA\_bn = *1×13*

-0.4773 0.0001 0.0000 -0.0001 -0.0955 0.0000 -0.0682 ⋯

[iB\_a0, iB\_an, iB\_bn] = computeHarmonicsAmplitude(iB, wt, orderOfHarmonics)

iB\_a0 = 2.0000e-04

iB\_an = *1×13*

10-3 ×

-0.4000 0.4000 -0.4000 0.4000 -0.4000 0.4000 -0.4000 ⋯

iB\_bn = *1×13*

0.9547 0.0005 -0.0000 -0.0005 0.1910 0.0000 0.1363 ⋯

[iC\_a0, iC\_an, iC\_bn] = computeHarmonicsAmplitude(iC, wt, orderOfHarmonics)

iC\_a0 = 2.0000e-04

iC\_an = *1×13*

-0.8272 -0.0004 -0.0008 -0.0003 0.1650 0.0004 -0.1185 ⋯

iC\_bn = *1×13*

-0.4770 -0.0006 0.0000 0.0006 -0.0958 -0.0000 -0.0678 ⋯

iA\_rec = reconstructFromFourierSeries(iA\_a0, iA\_an, iA\_bn, wt, orderOfHarmonics);

iB\_rec = reconstructFromFourierSeries(iB\_a0, iB\_an, iB\_bn, wt, orderOfHarmonics);

iC\_rec = reconstructFromFourierSeries(iC\_a0, iC\_an, iC\_bn, wt, orderOfHarmonics);

figure

tiledlayout(3,1)

titleHandle = sgtitle("Phase Currents Reconstructed from Fourier Series");

titleHandle.FontSize = 12;

nexttile;

plot(wt\_degrees,iA\_rec, ...

"DisplayName", "Reconstructed", ...

"LineWidth",1.2, ...

"Marker","o")

hold on

plot(wt\_degrees,iA, ...

"DisplayName", "Original", ...

"LineWidth",1.2)

hold off

legend

subtitle("Phase A Current");

ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,iB\_rec,wt\_degrees,iB,"LineWidth",1.2)

subtitle("Phase B Current");

ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

nexttile;

plot(wt\_degrees,iC\_rec,wt\_degrees,iC,"LineWidth",1.2)

subtitle("Phase C Current");

ylabel("Current (A)");

xlabel("Spatial Angle (°)");

xticks(0:30:360)

ylim([-1 1])

grid

A diagram of a phase

Description automatically generated

### Phase MMF Represented by Fourier Series at Angles of Interest

iA\_rec\_angleOfInterest = iA\_rec(indices);

iB\_rec\_angleOfInterest = iB\_rec(indices);

iC\_rec\_angleOfInterest = iC\_rec(indices);

MMFA\_rec = NA\_rec.\*iA\_rec\_angleOfInterest'; % Transposed

MMFB\_rec = NB\_rec.\*iB\_rec\_angleOfInterest';

MMFC\_rec = NC\_rec.\*iC\_rec\_angleOfInterest';

### Total MMF Represented by Fourier Series at Angles of Interest

totalMMF\_rec = MMFA\_rec + MMFB\_rec + MMFC\_rec;

figure;

tiledlayout(3, 4, 'TileSpacing', 'Compact', 'Padding', 'Compact');

sgtitle('Total MMF at Different ωt Degrees');

titleHandle.FontSize = 13;

A diagram of a phase

Description automatically generated

for k = 1:length(angleOfInterest)

nexttile;

plot(wt\_degrees, totalMMF\_rec(k,:), 'LineWidth', 1.2);

hold on

plot(wt\_degrees, totalMMF(k,:), 'LineWidth', 0.8);

hold off

ylim([-2 2]);

yticks(-2:0.5:2)

grid on;

xlabel("Spatial Angle (°)");

ylabel("MMF (A·Turns)");

title(sprintf('ωt = %.0f°', angleOfInterest(k)))

end

A graph of different angles

Description automatically generated

### Table

A table with numbers and symbols

Description automatically generated

**Space Harmonics** (on the top horizontal axis): These refer to the harmonic components of the winding functions. The harmonics listed here include 1st, 3rd, 5th, 7th, 9th, 11th, 13th, and 15th order harmonics.

**Time Harmonics** (on the left vertical axis): These correspond to the harmonic components of the current in the winding. The harmonics listed include 1st, 3rd, 5th, 7th, 9th, 11th, 13th, and 15th order harmonics as well.

harmonic\_matrix = zeros(orderOfHarmonics(end));

diagnal\_harmonics = diag(NA\_an.\*iA\_an + NB\_an.\*iB\_an + NC\_an.\*iC\_an);

cross\_harmonics = NA\_an.\*iA\_an' + NB\_an.\*iB\_an' + NC\_an.\*iC\_an';

harmonic\_matrix(harmonic\_matrix==0) = cross\_harmonics(harmonic\_matrix==0)

harmonic\_matrix = *13×13*

-0.4553 0.0002 -0.0004 0.0002 0.0916 -0.0000 -0.0646 ⋯

-0.0004 -0.0000 -0.0000 -0.0000 0.0001 -0.0000 -0.0001

-0.0002 0.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000

-0.0004 -0.0000 -0.0000 -0.0000 0.0001 0.0000 -0.0001

0.0911 -0.0000 0.0001 -0.0000 -0.0183 -0.0000 0.0129

0.0000 0.0000 0.0000 -0.0000 0.0000 0.0000 -0.0000

-0.0650 0.0000 -0.0001 0.0000 0.0131 -0.0000 -0.0092

-0.0004 -0.0000 -0.0000 -0.0000 0.0001 -0.0000 -0.0001

-0.0002 0.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000

-0.0004 -0.0000 -0.0000 -0.0000 0.0001 0.0000 -0.0001

⋮

% space = [2 1];

% time = [1 2];

%

% % space.\*time' = [2 1].\*[1;2]

% space.\*time'

% % space.\*time = [2 1].\*[1 2] = [2 2]

% space.\*time

% I'm confident with this method

function [a0,an,bn] = computeHarmonicsAmplitude(expr,wt,orderOfHarmonics)

a0 = mean(expr);

% Initialization

an = zeros(size(orderOfHarmonics));

bn = zeros(size(orderOfHarmonics));

for n = 1:length(orderOfHarmonics)

% Amplitude of even harmonics (cos terms)

an(n) = 2\*mean(expr.\*cos(n\*wt));

% Amplitude of odd harmonics (sin terms)

bn(n) = 2\*mean(expr.\*sin(n\*wt));

end

end