

## plotKMZ60TransferCurves

Plot NXP KMZ60 characterization field transfer curves.

### Contents

---

- [Syntax](#)
- [Description](#)
- [Examples](#)
- [Input Arguments](#)
- [Output Arguments](#)
- [Requirements](#)
- [See Also](#)

### Syntax

---

`plotKMZ60TransferCurves()`

### Description

---

**plotKMZ60TransferCurves()** plot characterization field of KMZ 60 sensor.

### Examples

---

`plotKMZ60TransferCurves();`

### Input Arguments

---

None

### Output Arguments

---

None

### Requirements

---

- Other m-files: none
- Subfunctions: none
- MAT-files required: data/NXP\_KMZ60\_Characterization\_2020-12-03\_16-53-16-721.mat, data/config.mat

### See Also

---

- [plotKMZ60CharField](#)

Created on December 05. 2020 by Tobias Wulf. Copyright Tobias Wulf 2020.

```
function plotKMZ60TransferCurves()
    try
        % load dataset path and dataset content into function workspace
        load('config.mat', 'PathVariables');
        load(PathVariables.kmz60DatasetPath, 'Data', 'Info');
        close all;
    catch ME
        rethrow(ME)
    end

    % load needed data from dataset in to local variables for better handling %%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % get from user which field to investigate and limits for plateau
    fields = Info.SensorOutput.CosinusBridge.Determination;
    nFields = length(fields);
    fprintf('Choose 1 of %d fields ...\n', nFields);
    for i = 1:nFields
        fprintf('%s\t\t\t(%d)\n', fields{i}, i);
    end
end
```

```

end

iField = input('Choice: ');
field = fields{iField};
pl = input('Plateu limit in kA/m: ');

Vcos = Data.SensorOutput.CosinusBridge.(field);
Vsin = Data.SensorOutput.SinusBridge.(field);
gain = Info.SensorOutput.BridgeGain;
HxScale = Data.MagneticField.hx;
HyScale = Data.MagneticField.hy;
Hmin = Info.MagneticField.MinAmplitude;
Hmax = Info.MagneticField.MaxAmplitude;

% get unit strings from
kApm = Info.Units.MagneticFieldStrength;
mV = Info.Units.SensorOutputVoltage;

% get dataset infos and format strings to place in figures
% subtitle string for all figures
infoStr = join([Info.SensorManufacturer, Info.Sensor, Info.SensorTechnology, ...
    Info.SensorType, "Sensor Characterization Dataset."]);
dateStr = join(["Created on", Info.Created, "by", 'Thorben Sch\uthe', ...
    "and updated on", Info.Edited, "by", Info.Editor + "."]);

% clear dataset all loaded
clear Data Info;

% figure save path for different formats %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fName = sprintf("kmz60_transfer_curves_%s", field);
fPath = fullfile(PathVariables.saveFiguresPath, fName);
fSvgPath = fullfile(PathVariables.saveImagesPath, 'svg', fName);
fEpsPath = fullfile(PathVariables.saveImagesPath, 'eps', fName);
fPdfPath = fullfile(PathVariables.saveImagesPath, 'pdf', fName);

% define slices and limits to plot %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Hslice = 128; % hit ca. 0 kA/m
Hlims = [-pl pl];
mVpVlims = [-8 8];

% create figure for plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig = figure('Name', 'Transfer Curves', ...
    'NumberTitle', 'off', ...
    'WindowStyle', 'normal', ...
    'MenuBar', 'none', ...
    'ToolBar', 'none', ...
    'Units', 'centimeters', ...
    'OuterPosition', [0 0 33 30], ...
    'PaperType', 'a4', ...
    'PaperUnits', 'centimeters', ...
    'PaperOrientation', 'landscape', ...
    'PaperPositionMode', 'auto', ...
    'DoubleBuffer', 'on', ...
    'RendererMode', 'manual', ...
    'Renderer', 'painters');

tdl = tiledlayout(fig, 2, 2, ...
    'Padding', 'compact', ...
    'TileSpacing', 'compact');

title(tdl, sprintf('Transfer Curves: %s', field), ...
    'FontWeight', 'normal', ...
    'FontSize', 18, ...
    'FontName', 'Times', ...
    'Interpreter', 'latex');

subtitle(tdl, [infoStr; dateStr], ...
    'FontWeight', 'normal', ...

```

```

        'FontSize', 14, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

% set colormap
colormap('jet');

% cosinus bridge %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nexttile(1);
im = imagesc(HxScale, HyScale, Vcos);
set(gca, 'YDir', 'normal');
set(im, 'AlphaData', ~isnan(Vcos));
yticks(xticks);
axis square xy;
grid on;

% plot lines for slice to investigate
hold on;
yline(HyScale(Hslice), 'k:', 'LineWidth', 3);
hold off;

xlabel(sprintf('$H_x$ in %s', kApm), ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

ylabel(sprintf('$H_y$ in %s', kApm), ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

title('$V_{cos}(H_x,H_y)$', ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

% sinus bridge %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nexttile(2);
im = imagesc(HxScale, HyScale, Vsin);
set(gca, 'YDir', 'normal');
set(im, 'AlphaData', ~isnan(Vsin));
yticks(xticks);
axis square xy;
grid on;

% plot lines for slice to investigate
hold on;
xline(HxScale(Hslice), 'k:', 'LineWidth', 3);
hold off;

xlabel(sprintf('$H_x$ in %s', kApm), ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

ylabel(sprintf('$H_y$ in %s', kApm), ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...
        'Interpreter', 'latex');

title('$V_{sin}(H_x,H_y)$', ...
        'FontWeight', 'normal', ...
        'FontSize', 12, ...
        'FontName', 'Times', ...

```

[illegible]

```
if strcmp(yesno, 'y')
    savefig(fig, fPath);
    print(fig, fSvgPath, '-dsvg');
    print(fig, fEpsPath, '-depsc', '-tiff', '-loose');
    print(fig, fPdfPath, '-dpdf', '-loose', '-fillpage');
end
close(fig)

end
```