# plotTDKTransferCurves

Plot TDK TAS2141 characterization field transfer curves.

## Syntax

```
plotTDKTransferCurves()
```

## Description

**plotTDKTransferCurves()** plot characterization field of TDK sensor.

## Examples

```
plotTDKTransferCurves();
```

## Input Arguments

**None**

## Output Arguments

**None**

## Requirements

- Other m-files: none
- Subfunctions: none
- MAT-files required: data/TDK_TAS2141_Characterization_2020-10-22_18-12-16-827.mat, data/config.mat

## See Also

- plotTDKCharField

Created on December 05. 2020 by Tobias Wulf. 

```
function plotTDKTransferCurves()
    try
        % load dataset path and dataset content into function workspace
        load('config.mat', 'PathVariables');
        load(PathVariables.tdkDatasetPath, '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(%d)\n', fields{i}, i);
    end

    iField = 1; % input('Choice: ');
    field = fields{iField};
```

```
pl = 5; % 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("tdk_transfer_curves_%s", field);
fPath = fullfile(PathVariables.saveImagesPath, fName);

% define slices and limits to plot %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Hslice = 128; % hit ca. 0 kA/m
Hlims = [-pl pl];
mVpVlims = [-175 175];

% create figure for plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fig = figure('Name', 'Transfer Curves', 'OuterPosition', [0 0 33 30]);

tiledlayout(fig, 2, 2);

disp('Info:');
disp([infoStr; dateStr]);
disp('Title:');
fprintf('KMZ 60 Transfer Curves: %s\n', field);
disp(["a) Cosine Bridge Characteristic"; ...
      "b) Sine Bridge Characteristic"; ...
      "c) Transfer Curves for const. H_x = H_y = 0"]);

% set colormap
colormap('jet');

% cosinus bridge %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nexttile(1);
im = imagesc(HxScale, HyScale, Vcos);
set(gca, 'YDir', 'normal');
set(im, 'AlphaData', ~isnan(Vcos));
xticks(-20:10:20);
yticks(-20:10:20);
axis square xy;
```

```
% plot lines for slice to investigate
hold on;
yline(HyScale(Hslice), 'k:', 'LineWidth', 3.5);
plot(pl*cosd(0:360), pl*sind(0:360), 'k-.', 'LineWidth', 3.5);
hold off;

xlabel(sprintf('$H_x$ in %s', kApm));
ylabel(sprintf('$H_y$ in %s', kApm));
title(sprintf('a) $V_{cos}(H_x,H_y)$, Gain $ = %.1f$', gain));

% sinus bridge %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nexttile(2);
im = imagesc(HxScale, HyScale, Vsin);
set(gca, 'YDir', 'normal');
set(im, 'AlphaData', ~isnan(Vsin));
xticks(-20:10:20);
yticks(-20:10:20);
axis square xy;

% plot lines for slice to investigate
hold on;
xline(HxScale(Hslice), 'k:', 'LineWidth', 3.5);
plot(pl*cosd(0:360), pl*sind(0:360), 'k-.', 'LineWidth', 3.5);
hold off;

xlabel(sprintf('$H_x$ in %s', kApm));
ylabel(sprintf('$H_y$ in %s', kApm));
title(sprintf('b) $V_{sin}(H_x,H_y)$, Gain $ = %.1f$', gain));

% colorbar for both %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cb = colorbar;
cb.Label.String = sprintf('$V_{out}$ in %s', mV);
cb.TickLabelInterpreter = 'latex';
cb.Label.Interpreter = 'latex';
cb.Label.FontSize = 20;

% cosinus bridge sclices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nexttile([1 2]);
% slices
p = plot(HxScale, Vcos(Hslice,:), HyScale, Vsin(:, Hslice)');

% plateau limits
if pl > 0
    hold on;
    xline(Hlims(1), 'k-.', 'LineWidth', 3.5);
    xline(Hlims(2), 'k-.', 'LineWidth', 3.5);
    hold off;
end

legend(p, {sprintf('$V_{cos}(H_x,H_y)$, $H_y \\approx 0$ %s', kApm), ...
           sprintf('$V_{sin}(H_x,H_y)$, $H_x \\approx 0$ %s', kApm)},...
       'Location', 'SouthEast');
ylabel(sprintf('$V_{out}$ in %s', mV));
xlabel(sprintf('$H$ in %s', kApm));
title('c) Cosine and Sine Transfer Curves');
ylim(mVpVlims);
xlim([Hmin Hmax])
```

```
    % save results of figure %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     yesno = input('Save? [y/n]: ', 's');
%     if strcmp(yesno, 'y')
%         savefig(fig, fPath);
%         print(fig, fPath, '-dsvg');
%         print(fig, fPath, '-depsc', '-tiff', '-loose');
%         print(fig, fPath, '-dpdf', '-loose', '-fillpage');
%     end
%     close(fig)

end
```

---

*Published with MATLAB® R2020b*