generateConfigMat

Generate configuration mat-file wich contains reusable configuration to control the software or certain function parameters. Centralized collection of configuration. If it is certain configuration needed place it here.

Contents

Requirements

- Other m-files required: None
- Subfunctions: None
- MAT-files required: None

See Also

- save
- load
- matfile

Clean Up

Clear variables from workspace to build up a fresh new configuration workspace.

```
disp('Clean up workspace ...');
clearvars;
clc;
```

GeneralOptions

General options like formats for strings or date or anything else what has no special relation to a theme complex. Fix parameters.

```
disp('Set general options ...');
GeneralOptions = struct;
GeneralOptions.dateFormat = 'yyyy-mm-dd_HH-MM-SS-FFF';
```

Path Variables

Key path variables and directories, often used in functions or scripts. Collet the path in a struct for easier save the struct fields as variables to config.mat via -struct flag. Fix parameters.

```
disp('Create current project instance to gather information ...');

% create current project instance to retrieve root information
projectInstance = matlab.project.currentProject;

disp('Set path variables ...');
PathVariables =struct;

% project root path, needs to be recreated generic to work on
% different machines
PathVariables.rootPath = projectInstance.RootFolder;

% path to data folder, which contains datasets and config.mat
PathVariables.dataPath = fullfile(PathVariables.rootPath, 'data');

% path to TDK TAS2141 TMR angular sensor characterization dataset
PathVariables.tdkDatasetPath = fullfile(PathVariables.dataPath, ...
    'TDK_TAS2141_Characterization_2020-10-22_18-12-16-827.mat');

% path to TDK TAS2141 TMR angular sensor characterization dataset
PathVariables.kmz60DatasetPath = fullfile(PathVariables.dataPath, ...
    'NXP_KMZ60_Characterization_2020-12-03_16-53-16-721.mat');

% path to config file dataset
PathVariables.configPath = fullfile(PathVariables.dataPath, ...
    'config.mat');

% path to training dataset folder
PathVariables.trainingDataPath = fullfile(PathVariables.dataPath, ...
    'training');

% path to test dataset folder
PathVariables.testDataPath = fullfile(PathVariables.dataPath, ...
    'test');

% path to documentation and m-files only for documentation
PathVariables.docsPath = fullfile(PathVariables.rootPath, ...
    'docs');

% path to publish html documentation output directory, helptoc.xml location
PathVariables.publishHtmlPath = fullfile(PathVariables.docsPath, 'html');

% path to save plots as images svg, eps, png, etc.
PathVariables.saveImagesPath = fullfile(PathVariables.publishHtmlPath, ...
    'images');

% path to save matlab figures
PathVariables.saveFiguresPath = fullfile(PathVariables.publishHtmlPath, ...
    'figures');

% path to latex docs folder
PathVariables.latexDocsPath = fullfile(PathVariables.docsPath, ...
```

```
    'latex');

% path to latex Thesis Tobias Wulf (take care if comment in)
% PathVariables.thesisTobiasWulf = fullfile(PathVariables.latexDocsPath, ...
%     'BA_Thesis_Tobias_Wulf');

% path to docs export folder for Manual
PathVariables.exportPublishPath = fullfile(PathVariables.latexDocsPath, ...
    'Manual');

% path to style sheet for html documentation, Matlab provided style sheet
PathVariables.publishStyleSheetPath = fullfile(PathVariables.publishHtmlPath, ...
    'docsHtmlStyleSheet.xsl');

% path to documentation search database entries for Matlab help browser support
PathVariables.helpsearchPath = fullfile(PathVariables.publishHtmlPath, ...
    'helpsearch-v3');

% path to executable m-file scripts of the project
PathVariables.scriptsPath = fullfile(PathVariables.rootPath, 'scripts');

% path to source code files, function and class files
PathVariables.srcPath = fullfile(PathVariables.rootPath, 'src');
```

Publish Options

These are general options for documents to publish. They are passed to the matlab publish function via a struct where each option gets its own field. The option struct can be copied and adjusted for differing publish conditions in example for scripts, functions, and bare document m-files. Initialize the option struct with output format field name and field value and add further fields (options) with point value. Fix parameters.

```
disp('Set publish options struct for publish function ...');
PublishOptions = struct('format', 'html');
PublishOptions.outputDir = PathVariables.publishHtmlPath;
PublishOptions.stylesheet = PathVariables.publishStyleSheetPath;
PublishOptions.createThumbnail = false;
PublishOptions.figureSnapMethod = 'entireFigureWindow';
PublishOptions.imageFormat = 'png';
PublishOptions.maxHeight = [];
PublishOptions.maxWidth = [];
PublishOptions.useNewFigure = false;
PublishOptions.evalCode = false;
PublishOptions.catchError = true;
PublishOptions.codeToEvaluate = [];
PublishOptions.maxOutputLines = Inf;
PublishOptions.showCode = true;
```

Sensor Array Options

The options control the build up of the sensor array in geometry and techincal behavior. This means number of sensors in the array and its size in mm. The supply and offset voltage of each sensor which is needed for using the characterization which is normed in mV/V. These parameters should be fix during generation a pulk of training or test data sets. The simulation function does not covers vectors yet.

```
disp('Set sensor array option for geometry and behavior ...');
SensorArrayOptions = struct;
```

```
% Geometry of the sensor array current sensor array can be. Fix parameter.
% square - square sensor array with even distances to each sensor point
SensorArrayOptions.geometry = 'square';

% Sensor array square dimension. Fix parameter.
SensorArrayOptions.dimension = 8;

% Sensor array edge length in mm. Fix parameter.
SensorArrayOptions.edge = 2;

% Sensor array simulated supply voltage in volts. Fix parameter.
SensorArrayOptions.Vcc = 5;

% Sensor array simulated offset voltage for bridge outputs in volts. Fix
% paramter.
SensorArrayOptions.Voff = 2.5;

% Senor array voltage norm factor to recalculate norm bridge outputs to
% given supply voltage and offset voltage, current normin is mV/V which
% implements factor of 1e3. Fix paramter.
SensorArrayOptions.Vnorm = 1e3;
```

Dipole Options

Dipole options to calculate the magnetic field which stimulate the sensor array. The dipole is gained to sphere with additional z distance to the array by sphere radius. These parameters should be fix during generation a pulk of training or test data sets. The simulation function does not covers vectors yet.

```
disp('Set dipole options to calculate magnetic stimulus ...');
DipoleOptions = struct;

% Radius in mm of magnetic sphere in which the magnetic dipole is centered.
% So it can be seen as z-offset to the sensor array. Fix parameter.
DipoleOptions.sphereRadius = 2;

% H-field magnitude to multiply of generated and relative normed dipole
% H-fields, the norming is done in zero position of [0 0 z0 + sphere radius] for
% 0° due to the position of the magnetic moment [-1 0 0] x and y components
% are not relevant, norming without tilt. Magnitude in kA/m. The magnitude
% refers that the sphere magnet has this H-field magnitude in a certain distance
% z0 in example sphere with 2mm sphere radius has a H magnitude of 200kA/m in
% 5mm distance. Standard field strength for ferrite sphere magnets are between
% 180 and 200kA/m. Fix parameter.
DipoleOptions.H0mag = 200;

% Distance in zero position of the spherical magnet in which the imprinted
% H-field strength magnitude takes effect. Together with the sphere radius and
% and the imprinted field strength magnitude the distance in rest position
% characterizes the spherical magnet to later relative positions of the sensor
% array and generated dipole H-fields in rotation simulation. In mm. Fix
% parameter.
DipoleOptions.z0 = 1;

% Magnetic moment magnitude attach rotation to the dipole field at a
% certain position with x, y and z components. Choose a huge value to
% prevent numeric failures, by norming the factor is eleminated later. Fix
% parameter.
```

```
DipoleOptions.M0mag = 1e6;
```

Traning Options

Training options gives the software the needed information to generate training datasets by the sensor array simulation with a dipole manget as stimulus which pushed with an z offeset to a sphere.

```
disp('Set training options to generate dataset ...');
TrainingOptions = struct;

% Use case of options define what dataset it is and where to save resulting
% datasets by simulation function. Fix parameter.
TrainingOptions.useCase = 'Training';

% Sensor array relative position to dipole magnet as position vector with
% x, y and z posiotn in mm. Negative x for left shift, negative y for up
% shift and negative z to place the layer under the dipole decrease z to
% increase the distance. The z-position will be subtracted by dipole sphere
% radius in simulation. So there is an offset given by the sphere radius.
% Loop parameters.
TrainingOptions.xPos = [0,];
TrainingOptions.yPos = [0, -1, -2, -5];
TrainingOptions.zPos = [7,];

% Dipole tilt in z-axes in degree. Fix parameter.
TrainingOptions.tilt = 0;

% Resolution of rotaion in degree, use same resoultion in training and test
% datasets to have the ability to back reference the index to fullscale
% test data sets. In degree. Fix parameter.
TrainingOptions.angleRes = 0.5;

% Phase index applies a phase offset in the rotation, it is used as phase index
% to a down sampling to generate even distributed angles of a full scale
% rotation. Offset index of full rotation. In example a full scale rotation from
% 0° to 360° - angleRes returns 720 angles, if nAngles is set to 7 it returns 7
% angles [0, 51.5, 103, 154.5, 206, 257.5, 309]. To get a phase shift of 11° set
% phaseIndex to 22 a multiple of the resolution angleRes and get
% [11, 62.5, 114, 165.5, 217, 268.5, 320]. Must be positive integer. Fix
% parameter.
TrainingOptions.phaseIndex = 0;

% Number rotaion angles, even distribute between 0° and 360° with respect
% to the resolution, even down sampling. To generate full scale the number
% relatead to the resolution or fast generate but wrong number set it to 0 to
% generate full scale rotation too. Fix Parameter.
TrainingOptions.nAngles = 16;

% Charcterization datset to use in simulation. Current available datasets are
% TDK - for characterization dataset of TDK TAS2141 TMR sensor
% KMZ60 - for characterization dataset of NXP KMZ60 AMR sensor
TrainingOptions.BaseReference = 'TDK';

% Characteraztion field which should be load as refernce image from
% characterization data set, in TDK dataset are following fields. In the
% current dataset Rise has the widest linear plateau with a radius of ca.
% 8.5 kA/m. Fix parameter.
% Rise - Bridge outputs for rising stimulus amplituded
```

```
% Fall - Bridge outputs for falling stimulus amplitude
% All  - Superimposed bridge outputs
% Diff - Differentiated bridge outputs
TrainingOptions.BridgeReference = 'Rise';
```

Test Options

Test options gives the software the needed information to generate test datasets by the sensor array simulation with a dipole manget as stimulus which pushed with an z offeset to a sphere.

```
disp('Set test options to generate dataset ...');
TestOptions = struct;

% Use case of options define what dataset it is and where to save resulting
% datasets by simulation function. Fix Parameter.
TestOptions.useCase = 'Test';

% Sensor array relative position to dipole magnet as position vector with
% x, y and z posiotn in mm. Negative x for left shift, negative y for up
% shift and negative z to place the layer under the dipole decrease z to
% increase the distance. The z-position will be subtracted by dipole sphere
% radius in simulation. So there is an offset given by the sphere radius.
% Loop parameter.
TestOptions.xPos = [];
TestOptions.yPos = [];
TestOptions.zPos = [];

% Dipole tilt in z-axes in degree. Fix parameter.
TestOptions.tilt = 0;

% Resolution of rotaion in degree, use same resoultion in training and test
% datasets to have the ability to back reference the index to fullscale
% test data sets. In degree. Fix parameter.
TestOptions.angleRes = 0.5;

% Phase index applies a phase offset in the rotation, it is used as phase index
% to a down sampling to generate even distributed angles of a full scale
% rotation. Offset index of full rotation. In example a full scale rotation from
% 0° to 360° - angleRes returns 720 angles, if nAngles is set to 7 it returns 7
% angles [0, 51.5, 103, 154.5, 206, 257.5, 309]. To get a phase shift of 11° set
% phaseIndex to 22 a multiple of the resolution angleRes and get
% [11, 62.5, 114, 165.5, 217, 268.5, 320]. Must be positive integer. Fix
% parameter.
TestOptions.phaseIndex = 0;

% Number rotaion angles, even distribute between 0° and 360° with respect
% to the resolution, even down sampling. To generate full scale the number
% relatead to the resolution or fast generate but wrong number to 0 to
% generate full scale rotation. Fix parameter.
TestOptions.nAngles = 720;

% Charcterization datset to use in simulation. Current available datasets are
% TDK - for characterization dataset of TDK TAS2141 TMR sensor
% KMZ60 - for characterization dataset of NXP KMZ60 AMR sensor
TestOptions.BaseReference = 'TDK';

% Characteraztion field which should be load as refernce image from
% characterization data set, in TDK dataset are following fields. In the
```

```
% current dataset Rise has the widest linear plateau with a radius of ca.
% 8.5 kA/m. Fix parameter.
% Rise - Bridge outputs for rising stimulus amplituded
% Fall - Bridge outputs for falling stimulus amplitude
% All  - Superimposed bridge outputs
% Diff - Differentiated bridge outputs
TestOptions.BridgeReference = 'Rise';
```

Save Configuration

Save section wise each config part as struct to standalone variables in config.mat use newest save format with no compression. create config.mat with timestamp of creation

```
disp('Create config.mat ...');
timestamp = datestr(now, GeneralOptions.dateFormat);
save(PathVariables.configPath, ...
    'timestamp', ...
    'GeneralOptions', ...
    'PathVariables', ...
    'PublishOptions', ...
    'SensorArrayOptions', ...
    'DipoleOptions', ...
    'TrainingOptions', ...
    'TestOptions', ...
    '-v7.3', '-nocompression');
```

Published with MATLAB® R2020b