Załącznik I (kod programu)

```
classdef HHGUIApp < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                      matlab.ui.Figure
        N0_MainGridLayout             matlab.ui.container.GridLayout
        N0_MainPanel                  matlab.ui.container.Panel
        N0_GridLayout                 matlab.ui.container.GridLayout
        N0_SaveAllButton              matlab.ui.control.Button
        N0_BusyLampLabel              matlab.ui.control.Label
        N0_BusyLamp                   matlab.ui.control.Lamp
        N0_TabGroup                   matlab.ui.container.TabGroup
        N0_BEMDTab                    matlab.ui.container.Tab
        N1ToN3_GridLayout             matlab.ui.container.GridLayout
        N1_Panel                      matlab.ui.container.Panel
        N1_GridLayout                 matlab.ui.container.GridLayout
        N1_FileNameLabel              matlab.ui.control.Label
        N1_LoadButton                 matlab.ui.control.Button
        N1_BM3DPanel                  matlab.ui.container.Panel
        N1_BM3DGridLayout             matlab.ui.container.GridLayout
        N1_SigmaEditFieldLabel        matlab.ui.control.Label
        N1_SigmaEditField             matlab.ui.control.NumericEditField
        N1_BM3DButton                 matlab.ui.control.Button
        N2_Panel                      matlab.ui.container.Panel
        N2_GridLayout                 matlab.ui.container.GridLayout
        N2_CalculateButton            matlab.ui.control.Button
        N2_TypeDropDownLabel          matlab.ui.control.Label
        N2_TypeDropDown               matlab.ui.control.DropDown
        N2_CutoffModEditField         matlab.ui.control.NumericEditField
        N2_LastComponentLabel         matlab.ui.control.Label
        N3_Panel                      matlab.ui.container.Panel
        N3_GridLayout                 matlab.ui.container.GridLayout
        N0_HilbertTab                 matlab.ui.container.Tab
        N4ToN9_GridLayout             matlab.ui.container.GridLayout
        N4_LoadedImagePanel           matlab.ui.container.Panel
        N4_GridLayout                 matlab.ui.container.GridLayout
        N4_LoadButton                 matlab.ui.control.Button
        N4_FromBEMDButton             matlab.ui.control.Button
        N4_FileNameLabel              matlab.ui.control.Label
        N5_Panel                      matlab.ui.container.Panel
        N5_GridLayout                 matlab.ui.container.GridLayout
        N5_CalculateButton            matlab.ui.control.Button
        N5_PasteNextButton            matlab.ui.control.Button
        N5_SizeEditFieldLabel         matlab.ui.control.Label
        N5_SizeEditField              matlab.ui.control.NumericEditField
        N6_Panel                      matlab.ui.container.Panel
        N6_GridLayout                 matlab.ui.container.GridLayout
        N6_FilterPanel                matlab.ui.container.Panel
        N6_FilterGridLayout           matlab.ui.container.GridLayout
        N6_FilterDropDown              matlab.ui.control.DropDown
        N6_CalculateButton             matlab.ui.control.Button
        N6_ParameterEditFieldLabel     matlab.ui.control.Label
        N6_ParameterEditField          matlab.ui.control.NumericEditField
        N9_Panel                       matlab.ui.container.Panel
        N9_GridLayout                  matlab.ui.container.GridLayout
        N9_CalculateButton             matlab.ui.control.Button
        N8_Panel                       matlab.ui.container.Panel
        N8_GridLayout                  matlab.ui.container.GridLayout
        N7_Panel                       matlab.ui.container.Panel
        N7_GridLayout                  matlab.ui.container.GridLayout
        N7_CalculateButton             matlab.ui.control.Button
        N0_SurfaceFitTab               matlab.ui.container.Tab
        N10to11_GridLayout             matlab.ui.container.GridLayout
        N10toN11_Panel                 matlab.ui.container.Panel
        N10toN11_GridLayout            matlab.ui.container.GridLayout
        N11_CalculateButton            matlab.ui.control.Button
        N11_PlotResultBox              matlab.ui.control.CheckBox
        N11_PlotPhaseBox               matlab.ui.control.CheckBox
        N10_Panel                      matlab.ui.container.Panel
        N10_GridLayout                 matlab.ui.container.GridLayout
        N10_LoadButton                 matlab.ui.control.Button
        N10_FromHilbertButton          matlab.ui.control.Button
        N10_FileNameLabel              matlab.ui.control.Label
        N10_BorderCutPanel             matlab.ui.container.Panel
        N10_BorderCutGridLayout        matlab.ui.container.GridLayout
        N10_SizeEditFieldLabel         matlab.ui.control.Label
        N10_SizeEditField              matlab.ui.control.NumericEditField
        N10_CalculateBorderCutButton   matlab.ui.control.Button
        N10_DimensionsLabel            matlab.ui.control.Label
        N11_PlotSubtractionBox         matlab.ui.control.CheckBox
        N11_PlotsLabel                 matlab.ui.control.Label
        N11_SubtractionColorDropDown   matlab.ui.control.DropDown
        N11_PhaseColorDropDown         matlab.ui.control.DropDown
        N11_FitColorDropDown           matlab.ui.control.DropDown
        N11_ColormapsLabel             matlab.ui.control.Label
        N11_Panel                      matlab.ui.container.Panel
        N11_GridLayout                 matlab.ui.container.GridLayout
        N11_UIAxes                     matlab.ui.control.UIAxes
        N0_HelpTab                     matlab.ui.container.Tab
        N12_Label                      matlab.ui.control.Label
        N0_ColorbarsButton             matlab.ui.control.StateButton
        N0_AutoButton                  matlab.ui.control.Button
        N0_LoadAllButton               matlab.ui.control.Button
        N0_ReportButton                matlab.ui.control.Button
        N0_ColormapDropDownLabel       matlab.ui.control.Label
        N0_ColormapDropDown            matlab.ui.control.DropDown
        N0_ViewModeDropDownLabel       matlab.ui.control.Label
        N0_ViewModeDropDown            matlab.ui.control.DropDown
        N0_TooltipsButton              matlab.ui.control.StateButton
    end
    properties (Access = public)

        % GUI Object Arrays
        varSaveBtn = [];
        varStateBtn = [];
        varPanel = [];
        varGrid = [];
        varFigBtn = [];

        % Constants
        maxId = 10;
        maxMods = 14;

        % Data containers
        newObj = [];
        newData = {};
        nrOfMods = 0;


        % Last variable options for reporting
        lastModsComp = [];
        lastModsEx = [];
        lastFiltersName = [];
        lastFiltersNr = [];
        lastBM3DSigma = double.empty;
        lastOrientWinSize = double.empty;
        lastBorderCutSize = double.empty;
        lastDecompMode = double.empty;
        lastCutoffComp = double.empty;
    end

    methods (Access = public)

        % Sets up tooltips and text in Help tab
        function setupTips(app)
            app.N12_Label.Text = {'HHGUIApp is a MATLAB/AppDesigner
application which allows performing the Hilbert-Huang transform on
bidimensional images. '; ...
                                  'The application is divided into five parts:
upper panel and four tabs, placed in order of the algorithm (BEMD, Hilbert,
Surface Fit, Help). '; ...
                                  'Help regarding the first three tabs can be
seen by hovering over the corresponding tabs on the top left.'; ''; ...
                                  'Theory Summary'; 'The Hilbert-Huang
transform starts with BEMD, which stands for Bidimensional Empirical Mode
Decomposition. '; ...
                                  'The root of the algorithm was first
proposed by Huang et al in 1996, however there are currently several different
decomposition types which boost the speed of the original algorithm or expand
it for wider purposes. '; ...
                                  'The main idea behind all of those
representation is to decompose an image into its BIMFs (Bidimensional
Intrinsic Mode Function). '; ...
```

```
                                  'What it means is that the input image is 
divided into a set of images which values oscillate around 0, except for the 
last, which is monotonous. '; ...
                                  'The order of the BIMFs in the set is also 
important, because the images are sorted by descending frequencies. '; ...
                                  'This means that the first BIMF always has 
the highest frequencies, and thus might contain noise. Last BIMF has the 
lowest frequencies, and is practically monotonous. '; ...
                                  'By discarding the first BIMF (in case it is 
too noisy) and last BIMF (in case it is monotonous), we can then compose the 
image back together, this time without the unwanted components. '; ...
                                  ''; 'The above mentioned operation is 
essential for the Hilbert-Huang algorithm, because its condition requires that 
the input image has no trend. '; ...
                                  'The most basic idea behind Hilbert 
transform is to create an output image, which is numerically shifted by +-pi/2 
from the input image. '; ...
                                  'The process is enhanced by the creation of 
fringe orientation map, which also solves the problem of closed fringes. '; 
...
                                  'The general formula associated with fringe 
images is: I(x, y) = a(x, y) + b(x, y)cos( fi(x, y) ) + n(x, y), where I - 
intensity, a - background, fi - phase, n - noise, b - fringe intensity. '; ...
                                  'Through BEMD, we can remove both background 
and intensity. We are left with two unknown variables (b, fi). '; ...
                                  'The output of the Hilbert spiral transform 
creates another image, and thus solves the equation system. Usually, the 
searched variable is the phase (fi), which later gets unwrapped. '; ...
                                  ''; 'To further process the information, the 
unwrapped phase distribution is fitted to a surface and then subtracted from 
it. '; ...
                                  ''; 'Any questions regarding the use of this 
app can be asked by email, wokolis@gmail.com'; 'Application created by Mateusz 
Kolinski'};

            app.N2_TypeDropDownLabel.Tooltip = {'1. FABEMD d = min[min(d_max), 
min(d_min)]'; '2. FABEMD d = max[min(d_max), min(d_min)]'; ...
                                               '3. FABEMD d = min[max(d_max), 
max(d_min)]'; '4. FABEMD d = max[max(d_max), max(d_min)]'; ...
                                               '5. d = 
0.5(mean(d_max)+mean(d_min))'; '6. EFEMD'};
            app.N2_TypeDropDown.Tooltip = app.N2_TypeDropDownLabel.Tooltip;
            app.N0_TooltipsButton.Value = 1;
            N0_EnableTooltips(app);
        end

        % Loading files
        function [output, fileName] = loadFile(app)
            [fileName, filePath] = 
uigetfile({'*.mat;*.bmp;*.jpg;*.tif;*.tiff',...
                                             'Formats (mat, bmp, jpg, tif, 
tiff)'; ...
                                             '*.*', ...
```

```
                                            'All Formats'}, ...
                                            'Load Image');
            if filePath ~= 0
                fullFileName = strcat(filePath, fileName);
                [~, ~, ext] = fileparts(fullFileName);
                ext = ext(2:end);

                if ext == "mat"
                    dataStruct = uiimport(fullFileName);
                    if isempty(dataStruct) == 0
                        dataStruct = orderfields(dataStruct);
                        data = struct2cell(dataStruct);
                        output = cell2mat(data(1));
                    else
                        output = double.empty;
                    end
                elseif ext == "tiff" || ext == "tif"
                    tiffFile = Tiff(fullFileName, 'r');
                    output = read(tiffFile);
                    close(tiffFile);
                elseif ext == "bmp" || ext == "jpg"
                    output = imread(fullFileName);
                else
                    output = imread(fullFileName);
                end

                if(size(output,3) == 3)
                    output = rgb2gray(output);
                end
            else
                output = double.empty;
            end

            figure(app.UIFigure);
        end

        % Saving images
        function saveFile(app, event)
            lampControl(app, "on");

            data = app.newData{event.Source.UserData};
            if isempty(data) == 0
                [fileName, filePath] =
    uiputfile({'*.mat';'*.tif';'*.bmp';'*.jpg'}, 'Save Image');

                if filePath ~= 0
                    fullFileName = strcat(filePath, fileName);
                    [~, ~, fullExt] = fileparts(fullFileName);
                    ext = fullExt(2:end);

                    if ext == "mat"
                        save(fullFileName, 'data');
                    elseif ext == "tif"
```

```
                        if isa(data, "double")
                            if min(data(:)) > 0
                                data = uint8(data);
                            end
                        end

                        tiffFile = Tiff(fullFileName, 'w');
                        setTag(tiffFile, 'ImageLength', size(data, 1));
                        setTag(tiffFile, 'ImageWidth', size(data, 2));

setTag(tiffFile,'Photometric',Tiff.Photometric.MinIsBlack);
                        setTag(tiffFile,'Compression',Tiff.Compression.None);

                        if isa(data, 'uint8') || isa(data, 'int8')
                            setTag(tiffFile,'BitsPerSample',  8);
                        elseif isa(data, 'uint16') || isa(data, 'int16')
                            setTag(tiffFile,'BitsPerSample', 16);
                        elseif isa(data, 'uint32') || isa(data, 'int32') ||
isa(data, 'single')
                            setTag(tiffFile,'BitsPerSample', 32);
                        elseif isa(data, 'double')
                            setTag(tiffFile,'BitsPerSample', 64);
                        end

                        if isa(data, 'uint8') || isa(data, 'uint16') ||
isa(data, 'uint32') || isa(data, 'uint64')

setTag(tiffFile,'SampleFormat',Tiff.SampleFormat.UInt);
                        elseif isa(data, 'int8') || isa(data, 'int16') ||
isa(data, 'int32') || isa(data, 'int64')

setTag(tiffFile,'SampleFormat',Tiff.SampleFormat.Int);
                        elseif isa(data, 'double') || isa(data, 'single')

setTag(tiffFile,'SampleFormat',Tiff.SampleFormat.IEEEFP);
                        end

                        setTag(tiffFile,'SamplesPerPixel',1);

setTag(tiffFile,'PlanarConfiguration',Tiff.PlanarConfiguration.Chunky);
                        write(tiffFile, data);
                        close(tiffFile);
                    elseif ext == "bmp" || ext == "jpg"
                        imwrite(data, fullFileName);
                    end
                end

                figure(app.UIFigure);
            end

            lampControl(app, "off");
        end
        % Creates composition and exclusion images (N2, N3) from selected
components
        function recalculateBEMDOutput(app)
            app.newData{2} = zeros(size(app.newData{1}));
            app.newData{3} = zeros(size(app.newData{1}));
            app.lastModsComp = [];
            app.lastModsEx = [];

            for x = 1:app.nrOfMods
                if app.varStateBtn(x).Value == 1
                    app.newData{2} = app.newData{2} + app.newData{x +
app.maxId};
                    app.lastModsComp = [app.lastModsComp x];
                else
                    app.newData{3} = app.newData{3} + app.newData{x +
app.maxId};
                    app.lastModsEx = [app.lastModsEx x];
                end
            end

            paint(app, app.newData{2}, app.newObj(2));
            paint(app, app.newData{3}, app.newObj(3));
        end

        % Creates new figure window for corresponding GUI image
        function newFigure(app, event)
            lampControl(app, "on");

            figure
            data = app.newData{event.Source.UserData};

            if app.N0_ViewModeDropDown.Value == "imagesc"
                colormap(app.N0_ColormapDropDown.Value);
                imagesc(data);
                if app.N0_ColorbarsButton.Value == 1
                    colorbars
                end
            elseif app.N0_ViewModeDropDown.Value == "imshow"
                imshow(rescaleTo255(app, data));
            end

            lampControl(app, "off");
        end

        % Scales image to 0-255 range for viewing
        function output = rescaleTo255(~, input)
            output = uint8(255*mat2gray(input));
        end

        % Main function filling images with data
        function paint(app, data, container)
            if isempty(data) == 0
                if app.N0_ViewModeDropDown.Value == "imagesc"
```

```matlab
                    container.Visible = 1;
                    title(container, []);
                    xlabel(container, []);
                    ylabel(container, []);
                    container.XAxis.TickLabels = {};
                    container.YAxis.TickLabels = {};
                    tempImgR = imagesc(data, 'Parent', container, 'XData', [1
container.Position(3)], 'YData', [1 container.Position(4)]);
                    colormap(container, app.N0_ColormapDropDown.Value);
                    container.XLim = [1 tempImgR.XData(2)];
                    container.YLim = [1 tempImgR.YData(2)];

                    if app.N0_ColorbarsButton.Value == 1
                        colorbar(container);
                    else
                        colorbar(container, 'off');
                    end
                elseif app.N0_ViewModeDropDown.Value == "imshow"
                    container.Visible = 1;
                    container.ImageSource = cat(3, rescaleTo255(app, data),
rescaleTo255(app, data), rescaleTo255(app, data));
                end
            end
        end

        % Changing images during viewing mode change
        function change(app, container, id)
            data = reinstate(app, container, id);
            paint(app, data, app.newObj(id));
        end

        % Deleting and creating data container
        function data = reinstate(app, container, id)
            parent = container.Parent;
            row = container.Layout.Row;
            column = container.Layout.Column;
            data = app.newData{id};


            delete(container);

            if app.N0_ViewModeDropDown.Value == "imagesc"
                app.newObj(id) = uiaxes(parent);
            elseif app.N0_ViewModeDropDown.Value == "imshow"
                app.newObj(id) = uiimage(parent);
            end

            app.newObj(id).Layout.Row = row;
            app.newObj(id).Layout.Column = column;
            app.newObj(id).Visible = 0;
        end

        % Filtering function during orientation map smoothing
        function output = filtering(app, input, type, parameter)
            sinTheta = sin(input);
            cosTheta = cos(input);

            if type == "Gauss"
                sinFiltered = imgaussfilt(sinTheta, parameter);
                cosFiltered = imgaussfilt(cosTheta, parameter);
                app.lastFiltersName = [app.lastFiltersName "Gauss, Sigma = "];
                app.lastFiltersNr = [app.lastFiltersNr parameter];
            elseif type == "Median"
                sinFiltered = medfilt2(sinTheta, [parameter parameter]);
                cosFiltered = medfilt2(cosTheta, [parameter parameter]);
                app.lastFiltersName = [app.lastFiltersName "Median, Window
Size = "];
                app.lastFiltersNr = [app.lastFiltersNr parameter];
            elseif type == "Mean"
                sinFiltered = filter2(fspecial('average', parameter),
sinTheta/255);
                cosFiltered = filter2(fspecial('average', parameter),
cosTheta/255);
                app.lastFiltersName = [app.lastFiltersName "Mean, Window Size
= "];
                app.lastFiltersNr = [app.lastFiltersNr parameter];
            elseif type == "BM3D"
                upperLimitSin = max(sinTheta(:));
                lowerLimitSin = min(sinTheta(:));
                upperLimitCos = max(cosTheta(:));
                lowerLimitCos = min(cosTheta(:));
                [~, sinFiltered] = BM3D(1, rescale(sinTheta, 0, 1),
parameter);
                [~, cosFiltered] = BM3D(1, rescale(cosTheta, 0, 1),
parameter);
                sinFiltered = rescale(sinFiltered, lowerLimitSin,
upperLimitSin);
                cosFiltered = rescale(cosFiltered, lowerLimitCos,
upperLimitCos);
                app.lastFiltersName = [app.lastFiltersName "BM3D, Window Size
= "];
                app.lastFiltersNr = [app.lastFiltersNr parameter];
            end


            output = angle(cosFiltered + 1i*sinFiltered);
        end

        % Cutting uneven edges
        function output = cutUnevenEdges(~, data)
            sizeXY = size(data);


            if mod(sizeXY(1), 2) == 1
                data = data(1:end - 1, :);
            end
```

```matlab
    if mod(sizeXY(2), 2) == 1
        data = data(:, 1:end - 1);
    end

    output = data;
end

% Removing images during the start of a new analysis
function blankOutTab(app, tab)
    if tab == "BEMD" || tab =="all"
        startPos = 2;
        endPos = 3;
        app.lastBM3DSigma = double.empty;
        app.lastModsComp = [];
        app.lastModsEx = [];
        app.lastDecompMode = double.empty;
        app.lastCutoffComp = double.empty;
        app.nrOfMods = 0;
        app.N1_BM3DPanel.Title = "BM3D";
        for x = app.maxId + 1:app.maxId + app.maxMods
            reinstate(app, app.newObj(x), x);
            app.newData{x} = double.empty;
        end

        for x = 1:app.maxMods
            app.varPanel(x).Visible = 0;
        end
    end

    if tab == "Hilbert" || tab =="all"
        startPos = 5;
        endPos = 9;
        app.N4_FileNameLabel.Text = "File Name";
        app.lastOrientWinSize = double.empty;
        app.lastFiltersName = [];
        app.lastFiltersNr = [];
    end

    if tab == "Fit" || tab =="all"
        startPos = 1;
        endPos = 0;
        app.N10_FileNameLabel.Text = "File Name";
        app.N11_GridLayout.Visible = 0;
        app.newData{app.maxId + app.maxMods + 1} = double.empty;
        app.newData{app.maxId + app.maxMods + 2} = double.empty;
        app.newData{app.maxId + app.maxMods + 3} = double.empty;
        app.lastBorderCutSize = double.empty;
    end

    if tab == "HH"
        startPos = 7;
        endPos = 9;
        app.lastFiltersName = [];
```

```
                app.lastFiltersNr = [];
            end

            if tab == "unwrap"
                startPos = 9;
                endPos = 9;
            end

            if tab == "all"
                startPos = 1;
                endPos = 10;
            end

            for x = startPos:endPos
                reinstate(app, app.newObj(x), x);
                app.newData{x} = double.empty;
            end
        end

        % Controls the lamp light when calculations are being made
        function lampControl(app, mode)
            if mode == "on"
                app.N0_BusyLamp.Enable = 1;
                drawnow;
            elseif mode == "off"
                app.N0_BusyLamp.Enable = 0;
            end
        end

        function [coe, X, Y] = fitSurface(~, data)
            [height, width] = size(data);
            [X, Y] = meshgrid(1:width, 1:height);
            [xData, yData, zData] = prepareSurfaceData(X, Y, data);
            ft = fittype( 'a*x^2 + b*y^2 + c*x + d*y + e', 'independent',
{'x', 'y'}, 'dependent', 'z' );
            opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts.StartPoint = [1 1 1 1 1];
            [fitresult, ~] = fit( [xData, yData], zData, ft, opts );
            coe = coeffvalues(fitresult);
        end

        % Paints surfaces to N11
        function paintSurfaces(app)
            cmapPhase = colormap(app.N11_UIAxes,
app.N11_PhaseColorDropDown.Value);
            cmapFit = colormap(app.N11_UIAxes,
app.N11_FitColorDropDown.Value);
            cmapSubtraction = colormap(app.N11_UIAxes,
app.N11_SubtractionColorDropDown.Value);

            colormapMatrixSize = [];
            colormapMatrixContent = [];
            dataMatrixAll = [];
            if app.N11_PlotPhaseBox.Value == 1
                colormapMatrixSize = [colormapMatrixSize;size(cmapPhase, 1)];
                colormapMatrixContent = [colormapMatrixContent;cmapPhase];
                dataMatrixAll = [dataMatrixAll ; app.newData{app.maxId +
app.maxMods + 1}(:)];
            end
            if app.N11_PlotResultBox.Value == 1
                colormapMatrixSize = [colormapMatrixSize;size(cmapFit, 1)];
                colormapMatrixContent = [colormapMatrixContent;cmapFit];
                dataMatrixAll = [dataMatrixAll ; app.newData{app.maxId +
app.maxMods + 2}(:)];
            end
            if app.N11_PlotSubtractionBox.Value == 1
                colormapMatrixSize = [colormapMatrixSize;size(cmapSubtraction,
1)];
                colormapMatrixContent =
[colormapMatrixContent;cmapSubtraction];
                dataMatrixAll = [dataMatrixAll ; app.newData{app.maxId +
app.maxMods + 3}(:)];
            end

            highData = 0;
            if app.N11_PlotPhaseBox.Value == 1
                highData = app.newData{app.maxId + app.maxMods + 1};
            else
                if app.N11_PlotResultBox.Value == 1
                    highData = app.newData{app.maxId + app.maxMods + 2};
                else
                    highData = app.newData{app.maxId + app.maxMods + 3};
                end
            end

            elemNr = min(colormapMatrixSize);
            colormap(app.N11_UIAxes, colormapMatrixContent);
            cmin = min(dataMatrixAll);
            cmax = max(dataMatrixAll);
            C1 = min(elemNr, round((elemNr - 1)*(highData - cmin)/(cmax -
cmin)) + 1);
            C2 = elemNr + C1;
            C3 = elemNr + C2;

            plotSum = app.N11_PlotPhaseBox.Value + app.N11_PlotResultBox.Value
+ app.N11_PlotSubtractionBox.Value;

            if app.N11_PlotPhaseBox.Value == 1
                h(1) = surf(app.newData{app.maxId + app.maxMods + 1},
'Parent', app.N11_UIAxes, 'LineStyle', 'none');
                set(h(1),'CData',C1);
                if plotSum > 1
                    hold(app.N11_UIAxes, 'on');
                end
            end
            if app.N11_PlotResultBox.Value == 1
```

```
                h(2) = surf(app.newData{app.maxId + app.maxMods + 2},
'Parent', app.N11_UIAxes, 'LineStyle', 'none');
                set(h(2),'CData',C2);
                if plotSum > 1
                    hold(app.N11_UIAxes, 'on')
                end
            end
            if app.N11_PlotSubtractionBox.Value == 1
                h(3) = surf(app.newData{app.maxId + app.maxMods + 3},
'Parent', app.N11_UIAxes, 'LineStyle', 'none');
                set(h(3),'CData',C3);
            end
            if plotSum > 1
                hold(app.N11_UIAxes, 'off');
            end

            minCarn = 0;
            maxCarn = 0;
            if app.N11_PlotPhaseBox.Value == 1
                minCarn = C1;
                if app.N11_PlotResultBox.Value == 0 &&
app.N11_PlotSubtractionBox.Value == 0
                    maxCarn = C1;
                end
            end

            if app.N11_PlotResultBox.Value == 1
                if app.N11_PlotPhaseBox.Value == 0
                    minCarn = C2;
                end
                if app.N11_PlotSubtractionBox.Value == 0
                    maxCarn = C2;
                end
            end

            if app.N11_PlotSubtractionBox.Value == 1
                maxCarn = C3;
                if app.N11_PlotPhaseBox.Value == 0 &&
app.N11_PlotResultBox.Value == 0
                    minCarn = C3;
                end
            end


            caxis(app.N11_UIAxes, [min(minCarn(:)) max(maxCarn(:))])

            zlabel(app.N11_UIAxes, 'Phase Distribution');
            xlabel(app.N11_UIAxes, 'X');
            ylabel(app.N11_UIAxes, 'Y');
            title(app.N11_UIAxes, 'Surface Fitting of Unwrapped Phase');
            app.N11_GridLayout.Visible = 1;
            app.N11_UIAxes.Visible = 1;
        end
        % Custom callbacks for code clarity
        function createCustomCallbacks(app)
            app.N0_TooltipsButton.ValueChangedFcn = createCallbackFcn(app, 
@N0_EnableTooltips);
            app.N0_SaveAllButton.ButtonPushedFcn = createCallbackFcn(app, 
@N0_SaveAll);
            app.N0_AutoButton.ButtonPushedFcn = createCallbackFcn(app, 
@N0_Auto);
            app.N0_ColorbarsButton.ValueChangedFcn = createCallbackFcn(app, 
@N0_Colorbars);
            app.N0_ViewModeDropDown.ValueChangedFcn = createCallbackFcn(app, 
@N0_ChangeMode);
            app.N0_LoadAllButton.ButtonPushedFcn = createCallbackFcn(app, 
@N0_LoadAll);
            app.N0_ReportButton.ButtonPushedFcn = createCallbackFcn(app, 
@N0_Report);
            app.N0_ColormapDropDown.ValueChangedFcn = createCallbackFcn(app, 
@N0_Repaint);
            app.N1_LoadButton.ButtonPushedFcn = createCallbackFcn(app, 
@N1_Load);
            app.N1_BM3DButton.ButtonPushedFcn = createCallbackFcn(app, 
@N1_BM3D);
            app.N2_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N2_Calculate);
            app.N4_LoadButton.ButtonPushedFcn = createCallbackFcn(app, 
@N4_Load);
            app.N4_FromBEMDButton.ButtonPushedFcn = createCallbackFcn(app, 
@N4_FromBEMD);
            app.N5_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N5_Calculate);
            app.N5_PasteNextButton.ButtonPushedFcn = createCallbackFcn(app, 
@N5_PasteNext);
            app.N6_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N6_Filter);
            app.N7_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N7_Calculate);
            app.N9_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N9_Calculate);
            app.N10_LoadButton.ButtonPushedFcn = createCallbackFcn(app, 
@N10_Load);
            app.N10_FromHilbertButton.ButtonPushedFcn = createCallbackFcn(app, 
@N10_FromHilbert);
            app.N10_CalculateBorderCutButton.ButtonPushedFcn = 
createCallbackFcn(app, @N10_Calculate);
            app.N11_CalculateButton.ButtonPushedFcn = createCallbackFcn(app, 
@N11_Calculate);
        end

        % Enables or disables viewing of most tooltips
        function N0_EnableTooltips(app)
            lampControl(app, "on");
            if app.N0_TooltipsButton.Value == 1
                app.N0_BEMDTab.Tooltip = {'An input image is converted to 
grayscale.'; 'It can be initially filtered by BM3D algorithm.'; ...
                                          'Available types of decompositions 
and their short window size calculating algorithms are shown when hovered over 
the combo box.'; ...
                                          'The decomposition can be shortened 
by assigning the maximum number of IMFs, after which the rest is composed to 
create a trend. '; ...
                                          'IMFs can be excluded from the 
output sum by clicking on the state button "Include".'};
                app.N0_SaveAllButton.Tooltip = {'Lets the user save all images 
currently visible in the application from all the tabs into a .mat file. '; 
...
                                                'Save variables compatibile 
with "Load All" button.'};
                app.N0_BusyLamp.Tooltip = {'Lights up when there are 
calculations being made.'};
                app.N0_BusyLampLabel.Tooltip = app.N0_BusyLamp.Tooltip;
                app.N0_HilbertTab.Tooltip = {'The output of BEMD tab can be 
copied to the input of Hilbert tab via the .From BEMD. button.'; ...
                                             'The orientation map HAS to be 
pasted into the Smoothing panel via the "Paste Next" button.'; ...
                                             'It can also restart the 
smoothing process this way. '; ...
                                             'Smoothing can be applied by four 
filters: BEMD, Gaussian, Median or Mean.'; 'They can be combined without 
issue.'; ...
                                             'Quadrature Fringe Pattern panel 
serves only as information, it is not used further (as opposed to Phase 
panel).'};
                app.N0_SurfaceFitTab.Tooltip = {'The output of Hilbert tab can 
be copied to the input of Surface Fit tab via the "Copy from Hilbert 
Unwrapping" button.'; ...
                                                'That image can have its 
borders cut via the Border Cut panel.'; ...
                                                'The tick boxes on the down 
left determine which information is to be shown.'};
                app.N0_ColorbarsButton.Tooltip = {'State button, which 
switches between showing the colorbars on the images. Deactivated in "imshow" 
mode.'};
                app.N0_AutoButton.Tooltip = {'Automatically calculates all 
steps of the algorithm, using current parameters. '; ...
                                             'Requires the first image in BEMD 
to be loaded.'};
                app.N0_LoadAllButton.Tooltip = {'Lets the user load all images 
previously saved by "Save All" button from .mat file.'};
                app.N0_ReportButton.Tooltip = {'Generates a .pdf report, which 
contains all currently visible images, also containing parameters they were 
calculated with.'};
                app.N0_ViewModeDropDownLabel.Tooltip = {'Switches between 
viewing the images as a color maps (matlab function "imagesc") or as images 
with [0, 255] intensities ("imshow").'; ...
```

```
                                                                  'BIMFs in "imshow" are
rescaled to [0,255] range.'};
                app.N0_ViewModeDropDown.Tooltip =
app.N0_ViewModeDropDownLabel.Tooltip;
                app.N0_ColormapDropDown.Tooltip = {'Switches between available
colormaps. Deactivates during "imshow" mode.'};
                app.N0_ColormapDropDownLabel.Tooltip =
app.N0_ColormapDropDown.Tooltip;
                app.N1_SigmaEditField.Tooltip = {'Recommended values range
from 5 to 30.'};
                app.N1_SigmaEditFieldLabel.Tooltip =
app.N1_SigmaEditField.Tooltip;
                app.N5_SizeEditField.Tooltip = {'Recommended values range from
2 to 31.'};
                app.N5_SizeEditFieldLabel.Tooltip =
app.N5_SizeEditField.Tooltip;
            else
                app.N0_BEMDTab.Tooltip = {''};
                app.N0_SaveAllButton.Tooltip = {''};
                app.N0_BusyLamp.Tooltip = {''};
                app.N0_BusyLampLabel.Tooltip = {''};
                app.N0_HilbertTab.Tooltip = {''};
                app.N0_SurfaceFitTab.Tooltip = {''};
                app.N0_ColorbarsButton.Tooltip = {''};
                app.N0_AutoButton.Tooltip = {''};
                app.N0_LoadAllButton.Tooltip = {''};
                app.N0_ReportButton.Tooltip = {''};
                app.N0_ViewModeDropDownLabel.Tooltip = {''};
                app.N0_ViewModeDropDown.Tooltip = {''};
                app.N0_ColormapDropDown.Tooltip = {''};
                app.N0_ColormapDropDownLabel.Tooltip = {''};
                app.N1_SigmaEditField.Tooltip = {''};
                app.N1_SigmaEditFieldLabel.Tooltip = {''};
                app.N5_SizeEditField.Tooltip = {''};
                app.N5_SizeEditFieldLabel.Tooltip = {''};
            end

            lampControl(app, "off");
        end

        % Repaints all images (because their parameters were changed)
        function N0_Repaint(app)
            lampControl(app, "on");

            for x=1:app.maxId + app.maxMods
                change(app, app.newObj(x), x);
            end

            lampControl(app, "off");
        end

        % Automatic example of full program course
        function N0_Auto(app)
```

```
            lampControl(app, "on");

            if isempty(app.newData{1}) == 0
                N1_BM3D(app);
                N2_Calculate(app);
                app.varStateBtn(app.nrOfMods).Value = 0;
                recalculateBEMDOutput(app);
                app.N0_TabGroup.SelectedTab = app.N0_HilbertTab;
                N4_FromBEMD(app);
                N5_Calculate(app);
                N5_PasteNext(app);
                N6_Filter(app);
                N7_Calculate(app);
                N9_Calculate(app);
                app.N0_TabGroup.SelectedTab = app.N0_SurfaceFitTab;
                N10_FromHilbert(app);
                N10_Calculate(app);
                N11_Calculate(app);
            end

            lampControl(app, "off");
        end

        % Generating report of current analysis
        function N0_Report(app)
            lampControl(app, "on");

            import mlreportgen.report.*
            import mlreportgen.dom.*

            if ~exist(strcat(pwd, "\Reports"), 'dir')
                mkdir(strcat(pwd, "\Reports"));
            end

            addpath(genpath(strcat(pwd, "\Reports")));

            format shortg
            dateTime = fix(clock);

            rep = Report(strcat(pwd, '\Reports\HilbertHuangReport', '_',
int2str(dateTime(1)), '_', ...
                                 int2str(dateTime(2)), '_',
int2str(dateTime(3)), '_', ...
                                 int2str(dateTime(4)), '_',
int2str(dateTime(5)), '_', int2str(dateTime(6))), 'pdf');
            add(rep, TitlePage('Title', 'Hilbert-Huang Report', 'Author',
''));
            add(rep, TableOfContents);

            chapters = [Chapter('Title', 'BEMD') Chapter('Title', 'HVT')
Chapter('Title', 'Surface Fitting')];
            sections = [];
            figures = [];
            for x = 1:app.maxMods + app.maxId + 2
                sections = [sections Section()];
            end

            sections(1) = Section('Title', 'BEMD: Input Image and BM3D');
            sections(2) = Section('Title', 'BEMD: Composed Image');
            sections(3) = Section('Title', 'BEMD: Excluded Components');

            for a = 1:app.nrOfMods
                sections(a + 3) = Section('Title', ['BEMD: Component '
int2str(a)]);
            end


            sections(3+app.nrOfMods+1) = Section('Title', 'HVT: Input Image');
            sections(3+app.nrOfMods+2) = Section('Title', 'HVT: Orientation
Map mod(2pi)');
            sections(3+app.nrOfMods+3) = Section('Title', 'HVT: Orientation
Map mod(2pi) Smoothed');
            sections(3+app.nrOfMods+4) = Section('Title', 'HVT: Quadrature
Fringe Pattern');
            sections(3+app.nrOfMods+5) = Section('Title', 'HVT: Phase');
            sections(3+app.nrOfMods+6) = Section('Title', 'HVT: Unwrapped
Phase');
            sections(3+app.nrOfMods+7) = Section('Title', 'Surface Fitting:
Input Image and border cutting');
            sections(3+app.nrOfMods+8) = Section('Title', 'Surface Fitting:
Unwrapped Phase');
            sections(3+app.nrOfMods+9) = Section('Title', 'Surface Fitting:
Surface Fit');
            sections(3+app.nrOfMods+10) = Section('Title', 'Surface Fitting:
Subtraction');

            for x = 1:3
                if isempty(app.newData{x}) == 0
                    if app.N0_ViewModeDropDown.Value == "imagesc"
                        figures = [figures Figure(imagesc(app.newData{x}))];
                        colormap(app.N0_ColormapDropDown.Value);
                        if app.N0_ColorbarsButton.Value == 1
                            colorbar
                        end
                    else
                        figures = [figures Figure(imshow(app.newData{x}))];
                    end

                    add(sections(x), getImpl(figures(x), rep));
                else
                    figures = [figures Figure(imagesc(app.newData{x}))];
                end

                if x == 1
                    if app.N1_FileNameLabel.Text ~= "File Name"
```

```
                        add(sections(x), ['File Name: ' 
app.N1_FileNameLabel.Text]);
                    end

                    if isempty(app.lastBM3DSigma) == 0
                        add(sections(x), ['BM3D with Sigma = ' 
int2str(app.lastBM3DSigma)]);
                    end
                elseif x == 2
                    if app.lastDecompMode == 1
                        modeDescription = 'FABEMD d = min[min(d_max), 
min(d_min)]';
                    elseif app.lastDecompMode == 2
                        modeDescription = 'FABEMD d = max[min(d_max), 
min(d_min)]';
                    elseif app.lastDecompMode == 3
                        modeDescription = 'FABEMD d = min[max(d_max), 
max(d_min)]';
                    elseif app.lastDecompMode == 4
                        modeDescription = 'FABEMD d = max[max(d_max), 
max(d_min)]';
                    elseif app.lastDecompMode == 5
                        modeDescription = 'd = 0.5(mean(d_max)+mean(d_min))';
                    elseif app.lastDecompMode == 6
                        modeDescription = 'EFEMD';
                    end

                    if isempty(app.lastDecompMode) == 0
                        add(sections(x), ['Composition Mode: ' 
modeDescription]);
                    end

                    if isempty(app.lastCutoffComp) == 0
                        add(sections(x), ['Cutoff Component Number: ' 
int2str(app.lastCutoffComp)]);
                    end

                    if isempty(app.lastModsComp) == 0
                        add(sections(x), ['Composed from following components: 
' int2str(app.lastModsComp)]);
                    end
                elseif x == 3
                    if isempty(app.lastModsEx) == 0
                        add(sections(x), ['Composed from following components: 
' int2str(app.lastModsEx)]);
                    end
                end
            end

            for x = app.maxId + 1:app.maxId + app.nrOfMods
                if x == app.maxId + app.nrOfMods
                    sections(x - app.maxId + 3) = Section('Title', 'BEMD: 
Residuum');
```

```
                end

                if app.N0_ViewModeDropDown.Value == "imagesc"
                    figures = [figures Figure(imagesc(app.newData{x}))];
                    if app.N0_ColorbarsButton.Value == 1
                        colorbar
                    end
                else
                    figures = [figures Figure(imshow(app.newData{x}))];
                end

                add(sections(x - app.maxId + 3), getImpl(figures(x - app.maxId
+ 3), rep));
            end

            for x = 4:10
                if isempty(app.newData{x}) == 0
                    if app.N0_ViewModeDropDown.Value == "imagesc"
                        figures = [figures Figure(imagesc(app.newData{x}))];
                        if app.N0_ColorbarsButton.Value == 1
                            colorbar
                        end
                    else
                        figures = [figures Figure(imshow(app.newData{x}))];
                    end

                    add(sections(x + app.nrOfMods), getImpl(figures(x +
app.nrOfMods), rep));
                else
                    figures = [figures Figure(imagesc(app.newData{x}))];
                end

                if x == 4
                    if app.N4_FileNameLabel.Text ~= "File Name"
                        add(sections(x + app.nrOfMods), ['File Name: '
app.N4_FileNameLabel.Text]);
                    end
                elseif x == 5
                    if isempty(app.lastOrientWinSize) == 0
                        add(sections(x + app.nrOfMods), ['Orientation map
window size: ' int2str(app.lastOrientWinSize)]);
                    end
                elseif x == 6
                    if isempty(app.lastFiltersNr) == 0
                        for y = 1:size(app.lastFiltersNr, 2)
                            add(sections(x + app.nrOfMods), ['Filter: '
convertStringsToChars(app.lastFiltersName(y)) ' '
int2str(app.lastFiltersNr(y))]);
                        end
                    end
                elseif x == 10
                    if app.N10_FileNameLabel.Text ~= "File Name"
```

```
                        add(sections(x + app.nrOfMods), ['File Name: '
app.N10_FileNameLabel.Text]);
                    end
                    if isempty(app.lastBorderCutSize) == 0
                        add(sections(x + app.nrOfMods), ['Border Cutting
Pixels: ' int2str(app.lastBorderCutSize)]);
                    end
                end
            end

            for x = 1:3
                figures = [figures Figure(surf(app.newData{app.maxId +
app.maxMods + x}, 'LineStyle', 'none'))];

                if isempty(app.newData{app.maxId + app.maxMods + x}) == 0
                    add(sections(app.maxId + app.nrOfMods + x),
getImpl(figures(app.maxId + app.nrOfMods + x), rep));
                end
            end

            for x = 1:3 + app.nrOfMods
                add(chapters(1), sections(x));
            end

            for x = 4:9
                add(chapters(2), sections(x + app.nrOfMods));
            end

            for x = 10:13
                add(chapters(3), sections(x + app.nrOfMods));
            end

            add(rep, chapters(1));
            add(rep, chapters(2));
            add(rep, chapters(3));

            delete(gcf);
            close(rep);
            rptview(rep);

            lampControl(app, "off");
        end

        % Saving images of current analysis
        function N0_SaveAll(app)
            lampControl(app, "on");

            [fileName, filePath] = uiputfile('*.mat','Save All');
            if isempty(fileName) == 0 && isempty(filePath) == 0
                fullFileName = strcat(filePath, fileName);

                inputBEMD = app.newData{1};
                sumBIMF = app.newData{2};
                excluded = app.newData{3};
                inputHil = app.newData{4};
                orientMod2Pi = app.newData{5};
                orientMod2PiSmooth = app.newData{6};
                hilQuadrature = app.newData{7};
                hilPhase = app.newData{8};
                hilUnwrapPhase = app.newData{9};
                fitInput = app.newData{10};
                zFit = app.newData{app.maxId + app.maxMods + 2};
                zSubtraction = app.newData{app.maxId + app.maxMods + 3};
                BIMF1 = app.newData{app.maxId + 1};
                BIMF2 = app.newData{app.maxId + 2};
                BIMF3 = app.newData{app.maxId + 3};
                BIMF4 = app.newData{app.maxId + 4};
                BIMF5 = app.newData{app.maxId + 5};
                BIMF6 = app.newData{app.maxId + 6};
                BIMF7 = app.newData{app.maxId + 7};
                BIMF8 = app.newData{app.maxId + 8};
                BIMF9 = app.newData{app.maxId + 9};
                BIMF10 = app.newData{app.maxId + 10};
                BIMF11 = app.newData{app.maxId + 11};
                BIMF12 = app.newData{app.maxId + 12};
                BIMF13 = app.newData{app.maxId + 13};
                BIMF14 = app.newData{app.maxId + 14};
                numberOfMods = app.nrOfMods;
                save(fullFileName, 'inputBEMD', 'sumBIMF', 'excluded',
'inputHil', 'orientMod2Pi', ...
                                  'orientMod2PiSmooth', 'hilQuadrature',
'hilPhase', 'hilUnwrapPhase', ...
                                  'fitInput', 'zFit', 'zSubtraction',  ...
                                  'BIMF1', 'BIMF2', 'BIMF3', 'BIMF4', 'BIMF5',
'BIMF6', 'BIMF7', 'BIMF8', ...
                                  'BIMF9', 'BIMF10', 'BIMF11', 'BIMF12',
'BIMF13', 'BIMF14', 'numberOfMods');
            end

            figure(app.UIFigure);

            lampControl(app, "off");
        end

        % Loading images of current analysis
        function N0_LoadAll(app)
            lampControl(app, "on");

            [fileName, filePath] = uigetfile('*.mat','Save All');
            if isempty(fileName) == 0 && isempty(filePath) == 0
                fullFileName = strcat(filePath, fileName);
                variables = {'inputBEMD', 'sumBIMF', 'excluded', 'inputHil',
'orientMod2Pi', ...
                              'orientMod2PiSmooth', 'hilQuadrature',
'hilPhase', 'hilUnwrapPhase', ...
                              'fitInput', 'zFit', 'zSubtraction',  ...
                              'BIMF1', 'BIMF2', 'BIMF3', 'BIMF4', 'BIMF5',
'BIMF6', 'BIMF7', 'BIMF8', ...
                              'BIMF9', 'BIMF10', 'BIMF11', 'BIMF12', 'BIMF13',
'BIMF14', 'numberOfMods'};
                load(fullFileName, variables{:});

                blankOutTab(app, "all");

                if exist('numberOfMods','var') == 1
                    app.nrOfMods = numberOfMods;
                end
                if exist('inputBEMD','var') == 1
                    app.newData{1} = inputBEMD;
                end
                if exist('sumBIMF', 'var') == 1
                    app.newData{2} = sumBIMF;
                end
                if exist('excluded', 'var') == 1
                    app.newData{3} = excluded;
                end
                if exist('BIMF1', 'var') == 1
                    app.newData{app.maxId + 1} = BIMF1;
                end
                if exist('BIMF2', 'var') == 1
                    app.newData{app.maxId + 2} = BIMF2;
                end
                if exist('BIMF3', 'var') == 1
                    app.newData{app.maxId + 3} = BIMF3;
                end
                if exist('BIMF4', 'var') == 1
                    app.newData{app.maxId + 4} = BIMF4;
                end
                if exist('BIMF5', 'var') == 1
                    app.newData{app.maxId + 5} = BIMF5;
                end
                if exist('BIMF6', 'var') == 1
                    app.newData{app.maxId + 6} = BIMF6;
                end
                if exist('BIMF7', 'var') == 1
                    app.newData{app.maxId + 7} = BIMF7;
                end
                if exist('BIMF8', 'var') == 1
                    app.newData{app.maxId + 8} = BIMF8;
                end
                if exist('BIMF9', 'var') == 1
                    app.newData{app.maxId + 9} = BIMF9;
                end
                if exist('BIMF10', 'var') == 1
                    app.newData{app.maxId + 10} = BIMF10;
                end
                if exist('BIMF11', 'var') == 1
                    app.newData{app.maxId + 11} = BIMF11;
                end
                if exist('BIMF12', 'var') == 1
                    app.newData{app.maxId + 12} = BIMF12;
                end
                if exist('BIMF13', 'var') == 1
                    app.newData{app.maxId + 13} = BIMF13;
                end
                if exist('BIMF14', 'var') == 1
                    app.newData{app.maxId + 14} = BIMF14;
                end
                if exist('inputHil', 'var') == 1
                    app.newData{4} = inputHil;
                end
                if exist('orientMod2Pi', 'var') == 1
                    app.newData{5} = orientMod2Pi;
                end
                if exist('orientMod2PiSmooth', 'var') == 1
                    app.newData{6} = orientMod2PiSmooth;
                end
                if exist('hilQuadrature', 'var') == 1
                    app.newData{7} = hilQuadrature;
                end
                if exist('hilPhase', 'var') == 1
                    app.newData{8} = hilPhase;
                end
                if exist('hilUnwrapPhase', 'var') == 1
                    app.newData{9} = hilUnwrapPhase;
                end
                if exist('fitInput', 'var') == 1
                    app.newData{10} = fitInput;
                end
                if exist('zFit', 'var') == 1 && exist('fitInput', 'var') == 1
&& exist('zSubtraction', 'var') ==1
                    app.newData{app.maxId + app.maxMods + 1} = fitInput;
                    app.newData{app.maxId + app.maxMods + 2} = zFit;
                    app.newData{app.maxId + app.maxMods + 3} = zSubtraction;
                    paintSurfaces(app);
                end

                for x=1:app.maxMods + app.maxId
                    paint(app, app.newData{x}, app.newObj(x));
                end

                for x=1:app.nrOfMods
                    app.varPanel(x).Visible = 1;
                end
            end


            figure(app.UIFigure);

            lampControl(app, "off");
        end
```

```
% Viewing colorbars
function N0_Colorbars(app)
    lampControl(app, "on");

    for x = 1:app.maxMods + app.maxId
        change(app, app.newObj(x), x);
    end

    lampControl(app, "off");
end

% Changing viewing mode
function N0_ChangeMode(app)
    lampControl(app, "on");

    if app.N0_ViewModeDropDown.Value == "imshow"
        app.N0_ColormapDropDown.Enable = 0;
        app.N0_ColorbarsButton.Enable = 0;
    else
        app.N0_ColormapDropDown.Enable = 1;
        app.N0_ColorbarsButton.Enable = 1;
    end

    for x=1:app.maxId + app.maxMods
        change(app, app.newObj(x), x);
    end

    lampControl(app, "off");
end

% Loading N1 image
function N1_Load(app)
    lampControl(app, "on");

    [app.newData{1}, fileName] = loadFile(app);

    if isempty(fileName) == 0 && isempty(app.newData{1}) == 0
        blankOutTab(app, "BEMD");
        app.N1_FileNameLabel.Text = fileName;
        app.newData{1} = cutUnevenEdges(app, app.newData{1});
        paint(app, app.newData{1}, app.newObj(1));
    end

    lampControl(app, "off");
end

% BM3D on N1 image
function N1_BM3D(app)
    lampControl(app, "on");

    if isempty(app.newData{1}) == 0
        blankOutTab(app, "BEMD");
```

```
                upperLimit = max(app.newData{1}(:));
                lowerLimit = min(app.newData{1}(:));
                data = double(mat2gray(app.newData{1}));
                [~, app.newData{1}] = BM3D(1, data,
app.N1_SigmaEditField.Value);
                app.newData{1} = rescale(app.newData{1}, lowerLimit,
upperLimit);

                paint(app, app.newData{1}, app.newObj(1));
                app.N1_BM3DPanel.Title = "BM3D - Done";
                app.lastBM3DSigma = app.N1_SigmaEditField.Value;
            end

            lampControl(app, "off");
        end

        % Decomposition of N1 image into N2-N18
        function N2_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{1}) == 0
                if app.N2_CutoffModEditField.Value < app.maxMods
                    cutoffComponent = app.N2_CutoffModEditField.Value;
                else
                    cutoffComponent = app.maxMods;
                end

                varData = FABEMD1(app.newData{1},
str2double(app.N2_TypeDropDown.Value), cutoffComponent);
                app.nrOfMods = length(varData);
                app.lastDecompMode = str2double(app.N2_TypeDropDown.Value);
                app.lastCutoffComp = cutoffComponent;

                for x = 1:length(varData)
                    app.newData{app.maxId + x} = varData{x};
                end

                for x = 1:app.maxMods
                    if x <= app.nrOfMods
                        if x == app.nrOfMods
                            app.varPanel(x).Title = "Residuum";
                        else
                            app.varPanel(x).Title = "BIMF " + x;
                        end

                        app.varPanel(x).Visible = 1;
                        app.varStateBtn(x).Value = 1;
                        paint(app, app.newData{app.maxId + x},
app.newObj(app.maxId + x));
                    else
                        app.varPanel(x).Visible = 0;
                    end
                end
                recalculateBEMDOutput(app);
            end

            lampControl(app, "off");
        end

        % Loading of N4 image
        function N4_Load(app)
            lampControl(app, "on");

            [app.newData{4}, fileName] = loadFile(app);

            if isempty(fileName) == 0 && isempty(app.newData{4}) == 0
                blankOutTab(app, "Hilbert");
                app.N4_FileNameLabel.Text = fileName;
                app.newData{4} = cutUnevenEdges(app, app.newData{4});
                paint(app, app.newData{4}, app.newObj(4));
            end

            lampControl(app, "off");
        end

        % Loading of N4 image from N2
        function N4_FromBEMD(app)
            lampControl(app, "on");

            if isempty(app.newData{2}) == 0
                app.newData{4} = app.newData{2};
                blankOutTab(app, "Hilbert");
                app.N4_FileNameLabel.Text = "File Name";
                paint(app, app.newData{4}, app.newObj(4));
            end

            lampControl(app, "off");
        end

        % Orientation map mod2Pi of N4 (made from N20)
        function N5_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{4}) == 0
                thetaPi = FringeOrientation(app.newData{4},
app.N5_SizeEditField.Value);
                app.lastOrientWinSize = app.N5_SizeEditField.Value;
                thetaRand = double(Miguel_2D_unwrapper(single(2 * thetaPi)));
                app.newData{5} = angle(cos(thetaRand / 2) + 1i * sin(thetaRand
/ 2));
                paint(app, app.newData{5}, app.newObj(5));
            end

            lampControl(app, "off");
        end
        % Pasting N5 output into N6
        function N5_PasteNext(app)
            lampControl(app, "on");

            if isempty(app.newData{5}) == 0
                app.newData{6} = app.newData{5};
                blankOutTab(app, "HH");
                paint(app, app.newData{6}, app.newObj(6));
            end

            lampControl(app, "off");
        end

        % Filtering with 4 buttons/filters
        function N6_Filter(app)
            lampControl(app, "on");

            if isempty(app.newData{6}) == 0
                blankOutTab(app, "HH");
                app.newData{6} = filtering(app, app.newData{6},
app.N6_FilterDropDown.Value, app.N6_ParameterEditField.Value);
                paint(app, app.newData{6}, app.newObj(6));
            end

            lampControl(app, "off");
        end

        % Hilbert Spiral Transform of N4 and N6 into N7 and N8
        function N7_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{4}) == 0 && isempty(app.newData{6}) == 0
                blankOutTab(app, "unwrap");
                [~, ~, ~, s]=HVT(double(app.newData{4}));
                theta = angle(cos(app.newData{6}) - 1i * sin(app.newData{6}));
                app.newData{7} = imag(-1i * exp(1i * ((-theta))) .* s);
                paint(app, app.newData{7}, app.newObj(7));

                app.newData{8} = angle(double(app.newData{4}) + 1i *
double(app.newData{7}));
                paint(app, app.newData{8}, app.newObj(8));
            end

            lampControl(app, "off");
        end

        % Unwrapping of N8 into N9
        function N9_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{8}) == 0
```

```
                app.newData{9} =
double(Miguel_2D_unwrapper(single(app.newData{8})));
                paint(app, app.newData{9}, app.newObj(9));
            end

            lampControl(app, "off");
        end

        % Loading N10 image
        function N10_Load(app)
            lampControl(app, "on");

            [app.newData{10}, fileName] = loadFile(app);

            if isempty(fileName) == 0 && isempty(app.newData{10}) == 0
                blankOutTab(app, "Fit");
                app.N10_FileNameLabel.Text = fileName;
                paint(app, app.newData{10}, app.newObj(10));
                app.N10_DimensionsLabel.Text =
strcat(int2str(size(app.newData{10}, 1)), "x", int2str(size(app.newData{10},
2)));
            end

            lampControl(app, "off");
        end

        % Loading N10 image from N9
        function N10_FromHilbert(app)
            lampControl(app, "on");

            if isempty(app.newData{9}) == 0
                app.newData{10} = app.newData{9};
                app.N10_FileNameLabel.Text = "File Name";
                app.N10_DimensionsLabel.Text =
strcat(int2str(size(app.newData{10}, 1)), "x", int2str(size(app.newData{10},
2)));
                paint(app, app.newData{10}, app.newObj(10));
                blankOutTab(app, "Fit");
            end

            lampControl(app, "off");
        end

        % Cutting borders of N10
        function N10_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{10}) == 0
                length = app.N10_SizeEditField.Value;
                app.lastBorderCutSize = app.N10_SizeEditField.Value;

                if length ~= 0 && 2 * length < size(app.newData{10}, 1) && 2 *
length < size(app.newData{10}, 2)
```

```
                    app.newData{10} = app.newData{10}(1 + length:end - length,
1 + length:end - length);
                end

                paint(app, app.newData{10}, app.newObj(10));
                app.N10_DimensionsLabel.Text =
strcat(int2str(size(app.newData{10}, 1)), "x", int2str(size(app.newData{10},
2)));
            end

            lampControl(app, "off");
        end

        % Surface fitting of N10
        function N11_Calculate(app)
            lampControl(app, "on");

            if isempty(app.newData{10}) == 0
                if app.N11_PlotResultBox.Value == 0 &&
app.N11_PlotPhaseBox.Value == 0 && app.N11_PlotSubtractionBox.Value == 0
                    app.N11_GridLayout.Visible = 0;
                else
                    [coe, xFit, yFit] = fitSurface(app, app.newData{10});
                    app.newData{app.maxId + app.maxMods + 1} =
app.newData{10};
                    app.newData{app.maxId + app.maxMods + 2} = coe(1) * xFit
.^ 2 + coe(2) * yFit .^ 2 + coe(3) * xFit + coe(4) * yFit + coe(5);
                    app.newData{app.maxId + app.maxMods + 3} = app.newData{10}
- app.newData{app.maxId + app.maxMods + 2};

                    paintSurfaces(app);
                end
            end

            lampControl(app, "off");
        end
    end


    % Callbacks that handle component events
    methods (Access = private)


        % Code that executes after component creation
        function startupFcn(app)
            lampControl(app, "on");

            if ~exist(strcat(pwd, "\Reports"), 'dir')
                mkdir(strcat(pwd, "\Reports"));
            end

            addpath(genpath(strcat(pwd, "\Functions")));
            addpath(genpath(strcat(pwd, "\Reports")));
            app.newData = cell(1, app.maxId + app.maxMods + 3);
            createCustomCallbacks(app);


            RowPos = 11;
            ColPos = 1;

            app.varPanel = gobjects(1, app.maxMods);
            app.varGrid = gobjects(1, app.maxMods);
            app.varStateBtn = gobjects(1, app.maxMods);
            app.newObj = gobjects(1, app.maxMods + app.maxId);
            app.varFigBtn = gobjects(1, app.maxMods + app.maxId + 3);
            app.varSaveBtn = gobjects(1, app.maxMods + app.maxId + 3);

            for x = 1:app.maxMods + app.maxId
                if x > app.maxId
                    app.varPanel(x - app.maxId) =
uipanel(app.N1ToN3_GridLayout);
                    app.varPanel(x - app.maxId).Visible = 0;
                    app.varPanel(x - app.maxId).Layout.Row = [RowPos RowPos +
4];
                    app.varPanel(x - app.maxId).Layout.Column = [ColPos ColPos
+ 5];

                    ColPos = ColPos + 6;

                    if ColPos > 37
                        RowPos = 16;
                        ColPos = 1;
                    end

                    app.varGrid(x - app.maxId) = uigridlayout(app.varPanel(x -
app.maxId));
                    app.varGrid(x - app.maxId).ColumnWidth = {'1x', 50};
                    app.varGrid(x - app.maxId).RowHeight = {'1x', '1x', '1x'};

                    app.varStateBtn(x - app.maxId) = uibutton(app.varGrid(x -
app.maxId), 'state', 'ValueChangedFcn', @(btn,event)
recalculateBEMDOutput(app));
                    app.varStateBtn(x - app.maxId).Layout.Row = 1;
                    app.varStateBtn(x - app.maxId).Layout.Column = 2;
                    app.varStateBtn(x - app.maxId).Text = "Include";
                    app.varStateBtn(x - app.maxId).Value = 1;
                    app.varStateBtn(x - app.maxId).FontSize = 10;
                end


                if x == 1
                    container = app.N1_GridLayout;
                elseif x == 2
                    container = app.N2_GridLayout;
                elseif x == 3
                    container = app.N3_GridLayout;
                elseif x == 4
                    container = app.N4_GridLayout;
                elseif x == 5
                    container = app.N5_GridLayout;
                elseif x == 6
                    container = app.N6_GridLayout;
                elseif x == 7
                    container = app.N7_GridLayout;
                elseif x == 8
                    container = app.N8_GridLayout;
                elseif x == 9
                    container = app.N9_GridLayout;
                elseif x == 10
                    container = app.N10_GridLayout;
                elseif x > app.maxId
                    container = app.varGrid(x - app.maxId);
                end

                app.newObj(x) = uiaxes(container);
                app.newObj(x).Visible = 0;

                app.varSaveBtn(x) = uibutton(container, 'push',
'ButtonPushedFcn', @(btn, event) saveFile(app, event));
                app.varSaveBtn(x).UserData = x;
                app.varSaveBtn(x).Text = "Save";


                if (x > app.maxId)
                    app.varSaveBtn(x).Layout.Row = 3;
                    app.varSaveBtn(x).Layout.Column = 2;
                else
                    app.varSaveBtn(x).Layout.Column = 5;

                    if x == 10
                        app.varSaveBtn(x).Layout.Row = 8;
                        app.varSaveBtn(x).Layout.Column = 3;
                    elseif x <= app.maxId
                        app.varSaveBtn(x).Layout.Row = 10;
                    elseif x > app.maxId
                        app.varSaveBtn(x).Layout.Row = 3;
                        app.varSaveBtn(x).Layout.Column = 1;
                    end
                end

                app.varFigBtn(x) = uibutton(container, 'push');
                app.varFigBtn(x).ButtonPushedFcn = createCallbackFcn(app,
@newFigure, true);
                app.varFigBtn(x).UserData = x;
                app.varFigBtn(x).Text = 'Figure';

                if x > app.maxId
                    app.varFigBtn(x).Layout.Row = 2;
                    app.varFigBtn(x).Layout.Column = 2;
                else
```

```
                app.varFigBtn(x).Layout.Column = 5;
                if x <= app.maxId && (x ~= 10)
                    app.varFigBtn(x).Layout.Row = 9;
                elseif x == 10
                    app.varFigBtn(x).Layout.Row = 7;
                    app.varFigBtn(x).Layout.Column = 3;
                end
            end

            if x == 10
                app.newObj(x).Layout.Row = [1 4];
                app.newObj(x).Layout.Column = [1 3];
            elseif x > app.maxId
                app.newObj(x).Layout.Row = [1 3];
                app.newObj(x).Layout.Column = 1;
            elseif x <= app.maxId
                app.newObj(x).Layout.Column = [1 4];
                app.newObj(x).Layout.Row = [1 10];
            end
        end

        container = app.N10toN11_GridLayout;
        for x = app.maxId + app.maxMods + 1:app.maxId + app.maxMods + 3
            app.varSaveBtn(x) = uibutton(container, 'push', 
'ButtonPushedFcn', @(btn, event) saveFile(app, event));
            app.varSaveBtn(x).Layout.Row = 17;
            app.varSaveBtn(x).Layout.Column = x - app.maxId - app.maxMods;
            app.varSaveBtn(x).UserData = x;
            app.varSaveBtn(x).Text = "Save";

            app.varFigBtn(x) = uibutton(container, 'push');
            app.varFigBtn(x).ButtonPushedFcn = createCallbackFcn(app, 
@newFigure, true);
            app.varFigBtn(x).Layout.Row = 18;
            app.varFigBtn(x).Layout.Column = x - app.maxId - app.maxMods;
            app.varFigBtn(x).UserData = x;
            app.varFigBtn(x).Text = 'Figure';
        end

        drawnow;
        app.UIFigure.WindowState = 'maximized';

        setupTips(app);


        lampControl(app, "off");
    end
end


% Component initialization
methods (Access = private)
```

```
        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Color = [0.902 0.9569 0.9686];
            app.UIFigure.Position = [100 100 1250 950];
            app.UIFigure.Name = 'Hilbert Huang';

            % Create N0_MainGridLayout
            app.N0_MainGridLayout = uigridlayout(app.UIFigure);
            app.N0_MainGridLayout.ColumnWidth = {'4.34x'};
            app.N0_MainGridLayout.RowHeight = {'13.61x'};

            % Create N0_MainPanel
            app.N0_MainPanel = uipanel(app.N0_MainGridLayout);
            app.N0_MainPanel.Title = 'Main Panel';
            app.N0_MainPanel.Layout.Row = 1;
            app.N0_MainPanel.Layout.Column = 1;

            % Create N0_GridLayout
            app.N0_GridLayout = uigridlayout(app.N0_MainPanel);
            app.N0_GridLayout.ColumnWidth = {55, 55, 75, 80, '1x', 55, 70, 70,
75, 55, 55, '1x', 35, 50, 65, 65, 80, '1x', 35, 20};
            app.N0_GridLayout.RowHeight = {19, '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x'};

            % Create N0_SaveAllButton
            app.N0_SaveAllButton = uibutton(app.N0_GridLayout, 'push');
            app.N0_SaveAllButton.Tooltip = {''};
            app.N0_SaveAllButton.Layout.Row = 1;
            app.N0_SaveAllButton.Layout.Column = 10;
            app.N0_SaveAllButton.Text = 'Save All';

            % Create N0_BusyLampLabel
            app.N0_BusyLampLabel = uilabel(app.N0_GridLayout);
            app.N0_BusyLampLabel.HorizontalAlignment = 'center';
            app.N0_BusyLampLabel.Layout.Row = 1;
            app.N0_BusyLampLabel.Layout.Column = 19;
            app.N0_BusyLampLabel.Text = 'Busy';

            % Create N0_BusyLamp
            app.N0_BusyLamp = uilamp(app.N0_GridLayout);
            app.N0_BusyLamp.Tooltip = {''};
            app.N0_BusyLamp.Layout.Row = 1;
            app.N0_BusyLamp.Layout.Column = 20;


            % Create N0_TabGroup
            app.N0_TabGroup = uitabgroup(app.N0_GridLayout);
            app.N0_TabGroup.AutoResizeChildren = 'off';
            app.N0_TabGroup.Layout.Row = [2 31];
            app.N0_TabGroup.Layout.Column = [1 20];


            % Create N0_BEMDTab
            app.N0_BEMDTab = uitab(app.N0_TabGroup);
            app.N0_BEMDTab.AutoResizeChildren = 'off';
            app.N0_BEMDTab.Tooltip = {''};
            app.N0_BEMDTab.Title = 'BEMD';


            % Create N1ToN3_GridLayout
            app.N1ToN3_GridLayout = uigridlayout(app.N0_BEMDTab);
            app.N1ToN3_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x'};
            app.N1ToN3_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x'};
            app.N1ToN3_GridLayout.Scrollable = 'on';


            % Create N1_Panel
            app.N1_Panel = uipanel(app.N1ToN3_GridLayout);
            app.N1_Panel.AutoResizeChildren = 'off';
            app.N1_Panel.Title = 'Loaded Image';
            app.N1_Panel.Layout.Row = [1 10];
            app.N1_Panel.Layout.Column = [1 14];


            % Create N1_GridLayout
            app.N1_GridLayout = uigridlayout(app.N1_Panel);
            app.N1_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N1_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};


            % Create N1_FileNameLabel
            app.N1_FileNameLabel = uilabel(app.N1_GridLayout);
            app.N1_FileNameLabel.HorizontalAlignment = 'center';
            app.N1_FileNameLabel.FontSize = 9;
            app.N1_FileNameLabel.Layout.Row = 2;
            app.N1_FileNameLabel.Layout.Column = 5;
            app.N1_FileNameLabel.Text = 'File Name';


            % Create N1_LoadButton
            app.N1_LoadButton = uibutton(app.N1_GridLayout, 'push');
            app.N1_LoadButton.Layout.Row = 1;
            app.N1_LoadButton.Layout.Column = 5;
            app.N1_LoadButton.Text = 'Load';

            % Create N1_BM3DPanel
            app.N1_BM3DPanel = uipanel(app.N1_GridLayout);
            app.N1_BM3DPanel.Title = 'BM3D';
            app.N1_BM3DPanel.Layout.Row = [3 6];
            app.N1_BM3DPanel.Layout.Column = 5;

            % Create N1_BM3DGridLayout
            app.N1_BM3DGridLayout = uigridlayout(app.N1_BM3DPanel);
            app.N1_BM3DGridLayout.ColumnWidth = {'1x'};
            app.N1_BM3DGridLayout.RowHeight = {'1x', '1x', '1x'};

            % Create N1_SigmaEditFieldLabel
            app.N1_SigmaEditFieldLabel = uilabel(app.N1_BM3DGridLayout);
            app.N1_SigmaEditFieldLabel.HorizontalAlignment = 'center';
            app.N1_SigmaEditFieldLabel.FontSize = 10;
            app.N1_SigmaEditFieldLabel.Layout.Row = 1;
            app.N1_SigmaEditFieldLabel.Layout.Column = 1;
            app.N1_SigmaEditFieldLabel.Text = 'Sigma';

            % Create N1_SigmaEditField
            app.N1_SigmaEditField = uieditfield(app.N1_BM3DGridLayout,
'numeric');
            app.N1_SigmaEditField.HorizontalAlignment = 'center';
            app.N1_SigmaEditField.Tooltip = {''};
            app.N1_SigmaEditField.Layout.Row = 2;
            app.N1_SigmaEditField.Layout.Column = 1;
            app.N1_SigmaEditField.Value = 30;

            % Create N1_BM3DButton
            app.N1_BM3DButton = uibutton(app.N1_BM3DGridLayout, 'push');
            app.N1_BM3DButton.FontSize = 10;
            app.N1_BM3DButton.Layout.Row = 3;
            app.N1_BM3DButton.Layout.Column = 1;
            app.N1_BM3DButton.Text = 'BM3D';

            % Create N2_Panel
            app.N2_Panel = uipanel(app.N1ToN3_GridLayout);
            app.N2_Panel.AutoResizeChildren = 'off';
            app.N2_Panel.Title = 'Sum of BIMFs';
            app.N2_Panel.Layout.Row = [1 10];
            app.N2_Panel.Layout.Column = [15 28];
            % Create N2_GridLayout
            app.N2_GridLayout = uigridlayout(app.N2_Panel);
            app.N2_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N2_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N2_CalculateButton
            app.N2_CalculateButton = uibutton(app.N2_GridLayout, 'push');
            app.N2_CalculateButton.FontSize = 11;
            app.N2_CalculateButton.Layout.Row = 7;
            app.N2_CalculateButton.Layout.Column = 5;
            app.N2_CalculateButton.Text = 'Calculate';

            % Create N2_TypeDropDownLabel
            app.N2_TypeDropDownLabel = uilabel(app.N2_GridLayout);
            app.N2_TypeDropDownLabel.HorizontalAlignment = 'center';
            app.N2_TypeDropDownLabel.Tooltip = {''};
            app.N2_TypeDropDownLabel.Layout.Row = 1;
            app.N2_TypeDropDownLabel.Layout.Column = 5;
            app.N2_TypeDropDownLabel.Text = 'Type';

            % Create N2_TypeDropDown
            app.N2_TypeDropDown = uidropdown(app.N2_GridLayout);
            app.N2_TypeDropDown.Items = {'1', '2', '3', '4', '5', '6'};
            app.N2_TypeDropDown.Tooltip = {''};
            app.N2_TypeDropDown.Layout.Row = 2;
            app.N2_TypeDropDown.Layout.Column = 5;
            app.N2_TypeDropDown.Value = '6';

            % Create N2_CutoffModEditField
            app.N2_CutoffModEditField = uieditfield(app.N2_GridLayout,
'numeric');
            app.N2_CutoffModEditField.Layout.Row = 5;
            app.N2_CutoffModEditField.Layout.Column = 5;
            app.N2_CutoffModEditField.Value = 14;

            % Create N2_LastComponentLabel
            app.N2_LastComponentLabel = uilabel(app.N2_GridLayout);
            app.N2_LastComponentLabel.HorizontalAlignment = 'center';
            app.N2_LastComponentLabel.FontSize = 8;
            app.N2_LastComponentLabel.Layout.Row = 4;
            app.N2_LastComponentLabel.Layout.Column = 5;
            app.N2_LastComponentLabel.Text = 'Last Component';

            % Create N3_Panel
            app.N3_Panel = uipanel(app.N1ToN3_GridLayout);
            app.N3_Panel.Title = 'Excluded Components';
            app.N3_Panel.Layout.Row = [1 10];
            app.N3_Panel.Layout.Column = [29 42];

            % Create N3_GridLayout
            app.N3_GridLayout = uigridlayout(app.N3_Panel);
            app.N3_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N3_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N0_HilbertTab
            app.N0_HilbertTab = uitab(app.N0_TabGroup);
            app.N0_HilbertTab.AutoResizeChildren = 'off';
            app.N0_HilbertTab.Tooltip = {''};
            app.N0_HilbertTab.Title = 'Hilbert';

            % Create N4ToN9_GridLayout
            app.N4ToN9_GridLayout = uigridlayout(app.N0_HilbertTab);
            app.N4ToN9_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x'};
            app.N4ToN9_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x'};

            % Create N4_LoadedImagePanel
            app.N4_LoadedImagePanel = uipanel(app.N4ToN9_GridLayout);
            app.N4_LoadedImagePanel.AutoResizeChildren = 'off';
            app.N4_LoadedImagePanel.Title = 'Loaded Image';
            app.N4_LoadedImagePanel.Layout.Row = [1 5];
            app.N4_LoadedImagePanel.Layout.Column = [1 7];

            % Create N4_GridLayout
            app.N4_GridLayout = uigridlayout(app.N4_LoadedImagePanel);
            app.N4_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N4_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N4_LoadButton
            app.N4_LoadButton = uibutton(app.N4_GridLayout, 'push');
            app.N4_LoadButton.Layout.Row = 1;
            app.N4_LoadButton.Layout.Column = 5;
            app.N4_LoadButton.Text = 'Load';

            % Create N4_FromBEMDButton
            app.N4_FromBEMDButton = uibutton(app.N4_GridLayout, 'push');
            app.N4_FromBEMDButton.FontSize = 10;
            app.N4_FromBEMDButton.Layout.Row = 3;
            app.N4_FromBEMDButton.Layout.Column = 5;
            app.N4_FromBEMDButton.Text = 'From BEMD';
            % Create N4_FileNameLabel
            app.N4_FileNameLabel = uilabel(app.N4_GridLayout);
            app.N4_FileNameLabel.HorizontalAlignment = 'center';
            app.N4_FileNameLabel.FontSize = 10;
            app.N4_FileNameLabel.Layout.Row = 2;
            app.N4_FileNameLabel.Layout.Column = 5;
            app.N4_FileNameLabel.Text = 'File Name';

            % Create N5_Panel
            app.N5_Panel = uipanel(app.N4ToN9_GridLayout);
            app.N5_Panel.Title = 'Orientation Map - Modulo 2Pi';
            app.N5_Panel.Layout.Row = [1 5];
            app.N5_Panel.Layout.Column = [8 14];

            % Create N5_GridLayout
            app.N5_GridLayout = uigridlayout(app.N5_Panel);
            app.N5_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N5_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N5_CalculateButton
            app.N5_CalculateButton = uibutton(app.N5_GridLayout, 'push');
            app.N5_CalculateButton.Layout.Row = 3;
            app.N5_CalculateButton.Layout.Column = 5;
            app.N5_CalculateButton.Text = 'Calculate';

            % Create N5_PasteNextButton
            app.N5_PasteNextButton = uibutton(app.N5_GridLayout, 'push');
            app.N5_PasteNextButton.FontSize = 9;
            app.N5_PasteNextButton.Layout.Row = 4;
            app.N5_PasteNextButton.Layout.Column = 5;
            app.N5_PasteNextButton.Text = 'Paste Next';

            % Create N5_SizeEditFieldLabel
            app.N5_SizeEditFieldLabel = uilabel(app.N5_GridLayout);
            app.N5_SizeEditFieldLabel.HorizontalAlignment = 'center';
            app.N5_SizeEditFieldLabel.Layout.Row = 1;
            app.N5_SizeEditFieldLabel.Layout.Column = 5;
            app.N5_SizeEditFieldLabel.Text = 'Size';

            % Create N5_SizeEditField
            app.N5_SizeEditField = uieditfield(app.N5_GridLayout, 'numeric');
            app.N5_SizeEditField.HorizontalAlignment = 'center';
            app.N5_SizeEditField.Tooltip = {''};
            app.N5_SizeEditField.Layout.Row = 2;
            app.N5_SizeEditField.Layout.Column = 5;
            app.N5_SizeEditField.Value = 5;


            % Create N6_Panel
            app.N6_Panel = uipanel(app.N4ToN9_GridLayout);
            app.N6_Panel.Title = 'Orientation Map - Modulo 2Pi - Smoothing';
            app.N6_Panel.Layout.Row = [1 5];
            app.N6_Panel.Layout.Column = [15 21];


            % Create N6_GridLayout
            app.N6_GridLayout = uigridlayout(app.N6_Panel);
            app.N6_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N6_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};


            % Create N6_FilterPanel
            app.N6_FilterPanel = uipanel(app.N6_GridLayout);
            app.N6_FilterPanel.Title = 'Filter';
            app.N6_FilterPanel.Layout.Row = [1 4];
            app.N6_FilterPanel.Layout.Column = 5;


            % Create N6_FilterGridLayout
            app.N6_FilterGridLayout = uigridlayout(app.N6_FilterPanel);
            app.N6_FilterGridLayout.ColumnWidth = {'1x'};
            app.N6_FilterGridLayout.RowHeight = {'1x', '1x', '1x', '1x'};


            % Create N6_FilterDropDown
            app.N6_FilterDropDown = uidropdown(app.N6_FilterGridLayout);
            app.N6_FilterDropDown.Items = {'BM3D', 'Gauss', 'Mean', 'Median'};
            app.N6_FilterDropDown.FontSize = 8;
            app.N6_FilterDropDown.Layout.Row = 1;
            app.N6_FilterDropDown.Layout.Column = 1;
            app.N6_FilterDropDown.Value = 'BM3D';


            % Create N6_CalculateButton
            app.N6_CalculateButton = uibutton(app.N6_FilterGridLayout,
'push');
            app.N6_CalculateButton.FontSize = 8;
            app.N6_CalculateButton.Layout.Row = 4;
            app.N6_CalculateButton.Layout.Column = 1;
            app.N6_CalculateButton.Text = 'Calculate';


            % Create N6_ParameterEditFieldLabel
            app.N6_ParameterEditFieldLabel = uilabel(app.N6_FilterGridLayout);
            app.N6_ParameterEditFieldLabel.HorizontalAlignment = 'center';
            app.N6_ParameterEditFieldLabel.FontSize = 9;
            app.N6_ParameterEditFieldLabel.Layout.Row = 2;
            app.N6_ParameterEditFieldLabel.Layout.Column = 1;
            app.N6_ParameterEditFieldLabel.Text = 'Parameter';

            % Create N6_ParameterEditField
            app.N6_ParameterEditField = uieditfield(app.N6_FilterGridLayout,
'numeric');
            app.N6_ParameterEditField.Layout.Row = 3;
            app.N6_ParameterEditField.Layout.Column = 1;
            app.N6_ParameterEditField.Value = 30;

            % Create N9_Panel
            app.N9_Panel = uipanel(app.N4ToN9_GridLayout);
            app.N9_Panel.Title = 'Unwrapped Phase';
            app.N9_Panel.Layout.Row = [6 10];
            app.N9_Panel.Layout.Column = [15 21];

            % Create N9_GridLayout
            app.N9_GridLayout = uigridlayout(app.N9_Panel);
            app.N9_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N9_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N9_CalculateButton
            app.N9_CalculateButton = uibutton(app.N9_GridLayout, 'push');
            app.N9_CalculateButton.Layout.Row = 1;
            app.N9_CalculateButton.Layout.Column = 5;
            app.N9_CalculateButton.Text = 'Calculate';

            % Create N8_Panel
            app.N8_Panel = uipanel(app.N4ToN9_GridLayout);
            app.N8_Panel.Title = 'Phase';
            app.N8_Panel.Layout.Row = [6 10];
            app.N8_Panel.Layout.Column = [8 14];

            % Create N8_GridLayout
            app.N8_GridLayout = uigridlayout(app.N8_Panel);
            app.N8_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N8_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N7_Panel
            app.N7_Panel = uipanel(app.N4ToN9_GridLayout);
            app.N7_Panel.Title = 'Quadrature Fringe Pattern';
            app.N7_Panel.Layout.Row = [6 10];
            app.N7_Panel.Layout.Column = [1 7];

            % Create N7_GridLayout
            app.N7_GridLayout = uigridlayout(app.N7_Panel);
            app.N7_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x', '1x'};
            app.N7_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x'};

            % Create N7_CalculateButton
            app.N7_CalculateButton = uibutton(app.N7_GridLayout, 'push');
            app.N7_CalculateButton.Layout.Row = 1;
            app.N7_CalculateButton.Layout.Column = 5;
            app.N7_CalculateButton.Text = 'Calculate';

            % Create N0_SurfaceFitTab
            app.N0_SurfaceFitTab = uitab(app.N0_TabGroup);
            app.N0_SurfaceFitTab.Tooltip = {''};
            app.N0_SurfaceFitTab.Title = 'Surface Fit';

            % Create N10to11_GridLayout
            app.N10to11_GridLayout = uigridlayout(app.N0_SurfaceFitTab);
            app.N10to11_GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x'};
            app.N10to11_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x'};

            % Create N10toN11_Panel
            app.N10toN11_Panel = uipanel(app.N10to11_GridLayout);
            app.N10toN11_Panel.Title = 'Options';
            app.N10toN11_Panel.Layout.Row = [1 15];
            app.N10toN11_Panel.Layout.Column = [1 3];

            % Create N10toN11_GridLayout
            app.N10toN11_GridLayout = uigridlayout(app.N10toN11_Panel);
            app.N10toN11_GridLayout.ColumnWidth = {'1x', '1x', '1x'};
            app.N10toN11_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x', '1x',
'1x'};

            % Create N11_CalculateButton
            app.N11_CalculateButton = uibutton(app.N10toN11_GridLayout,
'push');
            app.N11_CalculateButton.Layout.Row = 19;
            app.N11_CalculateButton.Layout.Column = [1 3];
            app.N11_CalculateButton.Text = 'Calculate';

            % Create N11_PlotResultBox
            app.N11_PlotResultBox = uicheckbox(app.N10toN11_GridLayout);
            app.N11_PlotResultBox.Text = 'Fit';
            app.N11_PlotResultBox.FontSize = 11;
            app.N11_PlotResultBox.Layout.Row = 14;
            app.N11_PlotResultBox.Layout.Column = 2;


            % Create N11_PlotPhaseBox
            app.N11_PlotPhaseBox = uicheckbox(app.N10toN11_GridLayout);
            app.N11_PlotPhaseBox.Text = 'Phase';
            app.N11_PlotPhaseBox.FontSize = 11;
            app.N11_PlotPhaseBox.Layout.Row = 14;
            app.N11_PlotPhaseBox.Layout.Column = 1;


            % Create N10_Panel
            app.N10_Panel = uipanel(app.N10toN11_GridLayout);
            app.N10_Panel.Title = 'Current Input';
            app.N10_Panel.Layout.Row = [1 11];
            app.N10_Panel.Layout.Column = [1 3];


            % Create N10_GridLayout
            app.N10_GridLayout = uigridlayout(app.N10_Panel);
            app.N10_GridLayout.ColumnWidth = {'1x', '1x', '1x'};
            app.N10_GridLayout.RowHeight = {'1x', '1x', '1x', '1x', '1x',
'1x', '1x', '1x'};


            % Create N10_LoadButton
            app.N10_LoadButton = uibutton(app.N10_GridLayout, 'push');
            app.N10_LoadButton.Layout.Row = 5;
            app.N10_LoadButton.Layout.Column = 2;
            app.N10_LoadButton.Text = 'Load';


            % Create N10_FromHilbertButton
            app.N10_FromHilbertButton = uibutton(app.N10_GridLayout, 'push');
            app.N10_FromHilbertButton.FontSize = 10;
            app.N10_FromHilbertButton.Layout.Row = 5;
            app.N10_FromHilbertButton.Layout.Column = 1;
            app.N10_FromHilbertButton.Text = 'From Hilbert';


            % Create N10_FileNameLabel
            app.N10_FileNameLabel = uilabel(app.N10_GridLayout);
            app.N10_FileNameLabel.HorizontalAlignment = 'center';
            app.N10_FileNameLabel.Layout.Row = 5;
            app.N10_FileNameLabel.Layout.Column = 3;
            app.N10_FileNameLabel.Text = 'File Name';


            % Create N10_BorderCutPanel
            app.N10_BorderCutPanel = uipanel(app.N10_GridLayout);
            app.N10_BorderCutPanel.Title = 'Border Cut';
            app.N10_BorderCutPanel.Layout.Row = [6 8];
            app.N10_BorderCutPanel.Layout.Column = [1 2];
            % Create N10_BorderCutGridLayout
            app.N10_BorderCutGridLayout =
uigridlayout(app.N10_BorderCutPanel);

            % Create N10_SizeEditFieldLabel
            app.N10_SizeEditFieldLabel = uilabel(app.N10_BorderCutGridLayout);
            app.N10_SizeEditFieldLabel.HorizontalAlignment = 'right';
            app.N10_SizeEditFieldLabel.Layout.Row = 1;
            app.N10_SizeEditFieldLabel.Layout.Column = 1;
            app.N10_SizeEditFieldLabel.Text = 'Size';

            % Create N10_SizeEditField
            app.N10_SizeEditField = uieditfield(app.N10_BorderCutGridLayout,
'numeric');
            app.N10_SizeEditField.Layout.Row = 1;
            app.N10_SizeEditField.Layout.Column = 2;
            app.N10_SizeEditField.Value = 10;

            % Create N10_CalculateBorderCutButton
            app.N10_CalculateBorderCutButton =
uibutton(app.N10_BorderCutGridLayout, 'push');
            app.N10_CalculateBorderCutButton.Layout.Row = 2;
            app.N10_CalculateBorderCutButton.Layout.Column = [1 2];
            app.N10_CalculateBorderCutButton.Text = 'Calculate';

            % Create N10_DimensionsLabel
            app.N10_DimensionsLabel = uilabel(app.N10_GridLayout);
            app.N10_DimensionsLabel.HorizontalAlignment = 'center';
            app.N10_DimensionsLabel.Layout.Row = 6;
            app.N10_DimensionsLabel.Layout.Column = 3;
            app.N10_DimensionsLabel.Text = 'Dimensions';

            % Create N11_PlotSubtractionBox
            app.N11_PlotSubtractionBox = uicheckbox(app.N10toN11_GridLayout);
            app.N11_PlotSubtractionBox.Text = 'Subtraction';
            app.N11_PlotSubtractionBox.FontSize = 11;
            app.N11_PlotSubtractionBox.Layout.Row = 14;
            app.N11_PlotSubtractionBox.Layout.Column = 3;
            app.N11_PlotSubtractionBox.Value = true;

            % Create N11_PlotsLabel
            app.N11_PlotsLabel = uilabel(app.N10toN11_GridLayout);
            app.N11_PlotsLabel.HorizontalAlignment = 'center';
            app.N11_PlotsLabel.Layout.Row = 13;
            app.N11_PlotsLabel.Layout.Column = [1 3];
            app.N11_PlotsLabel.Text = 'Plots';
            % Create N11_SubtractionColorDropDown
            app.N11_SubtractionColorDropDown = 
uidropdown(app.N10toN11_GridLayout);
            app.N11_SubtractionColorDropDown.Items = {'parula', 'jet', 'hsv', 
'hot', 'cool', 'spring', 'summer', 'autumn', 'winter'};
            app.N11_SubtractionColorDropDown.FontSize = 11;
            app.N11_SubtractionColorDropDown.Layout.Row = 16;
            app.N11_SubtractionColorDropDown.Layout.Column = 3;
            app.N11_SubtractionColorDropDown.Value = 'parula';

            % Create N11_PhaseColorDropDown
            app.N11_PhaseColorDropDown = uidropdown(app.N10toN11_GridLayout);
            app.N11_PhaseColorDropDown.Items = {'parula', 'jet', 'hsv', 'hot', 
'cool', 'spring', 'summer', 'autumn', 'winter'};
            app.N11_PhaseColorDropDown.FontSize = 11;
            app.N11_PhaseColorDropDown.Layout.Row = 16;
            app.N11_PhaseColorDropDown.Layout.Column = 1;
            app.N11_PhaseColorDropDown.Value = 'hot';

            % Create N11_FitColorDropDown
            app.N11_FitColorDropDown = uidropdown(app.N10toN11_GridLayout);
            app.N11_FitColorDropDown.Items = {'parula', 'jet', 'hsv', 'hot', 
'cool', 'spring', 'summer', 'autumn', 'winter'};
            app.N11_FitColorDropDown.FontSize = 11;
            app.N11_FitColorDropDown.Layout.Row = 16;
            app.N11_FitColorDropDown.Layout.Column = 2;
            app.N11_FitColorDropDown.Value = 'summer';

            % Create N11_ColormapsLabel
            app.N11_ColormapsLabel = uilabel(app.N10toN11_GridLayout);
            app.N11_ColormapsLabel.HorizontalAlignment = 'center';
            app.N11_ColormapsLabel.Layout.Row = 15;
            app.N11_ColormapsLabel.Layout.Column = [1 3];
            app.N11_ColormapsLabel.Text = 'Colormaps';

            % Create N11_Panel
            app.N11_Panel = uipanel(app.N10to11_GridLayout);
            app.N11_Panel.Title = 'Result';
            app.N11_Panel.Layout.Row = [1 15];
            app.N11_Panel.Layout.Column = [4 13];

            % Create N11_GridLayout
            app.N11_GridLayout = uigridlayout(app.N11_Panel);
            app.N11_GridLayout.ColumnWidth = {'1x'};
            app.N11_GridLayout.RowHeight = {'1x'};
```

```
% Create N11_UIAxes
app.N11_UIAxes = uiaxes(app.N11_GridLayout);
title(app.N11_UIAxes, 'Title')
xlabel(app.N11_UIAxes, 'X')
ylabel(app.N11_UIAxes, 'Y')
app.N11_UIAxes.PlotBoxAspectRatio = [1.25267665952891 1 1];
app.N11_UIAxes.Visible = 'off';
app.N11_UIAxes.Layout.Row = 1;
app.N11_UIAxes.Layout.Column = 1;


% Create N0_HelpTab
app.N0_HelpTab = uitab(app.N0_TabGroup);
app.N0_HelpTab.Title = 'Help';


% Create N12_Label
app.N12_Label = uilabel(app.N0_HelpTab);
app.N12_Label.Position = [10 14 1188 810];
app.N12_Label.Text = '';


% Create N0_ColorbarsButton
app.N0_ColorbarsButton = uibutton(app.N0_GridLayout, 'state');
app.N0_ColorbarsButton.Tooltip = {''};
app.N0_ColorbarsButton.Text = 'Colorbars';
app.N0_ColorbarsButton.Layout.Row = 1;
app.N0_ColorbarsButton.Layout.Column = 15;


% Create N0_AutoButton
app.N0_AutoButton = uibutton(app.N0_GridLayout, 'push');
app.N0_AutoButton.Tooltip = {''};
app.N0_AutoButton.Layout.Row = 1;
app.N0_AutoButton.Layout.Column = 13;
app.N0_AutoButton.Text = 'Auto';


% Create N0_LoadAllButton
app.N0_LoadAllButton = uibutton(app.N0_GridLayout, 'push');
app.N0_LoadAllButton.Tooltip = {''};
app.N0_LoadAllButton.Layout.Row = 1;
app.N0_LoadAllButton.Layout.Column = 11;
app.N0_LoadAllButton.Text = 'Load All';


% Create N0_ReportButton
app.N0_ReportButton = uibutton(app.N0_GridLayout, 'push');
app.N0_ReportButton.Tooltip = {''};
app.N0_ReportButton.Layout.Row = 1;
app.N0_ReportButton.Layout.Column = 14;
app.N0_ReportButton.Text = 'Report';
```

```
            % Create N0_ColormapDropDownLabel
            app.N0_ColormapDropDownLabel = uilabel(app.N0_GridLayout);
            app.N0_ColormapDropDownLabel.HorizontalAlignment = 'center';
            app.N0_ColormapDropDownLabel.Layout.Row = 1;
            app.N0_ColormapDropDownLabel.Layout.Column = 8;
            app.N0_ColormapDropDownLabel.Text = 'Colormap';


            % Create N0_ColormapDropDown
            app.N0_ColormapDropDown = uidropdown(app.N0_GridLayout);
            app.N0_ColormapDropDown.Items = {'parula', 'jet', 'hsv', 'hot',
'cool', 'spring', 'summer', 'autumn', 'winter'};
            app.N0_ColormapDropDown.Tooltip = {''};
            app.N0_ColormapDropDown.Layout.Row = 1;
            app.N0_ColormapDropDown.Layout.Column = 9;
            app.N0_ColormapDropDown.Value = 'parula';


            % Create N0_ViewModeDropDownLabel
            app.N0_ViewModeDropDownLabel = uilabel(app.N0_GridLayout);
            app.N0_ViewModeDropDownLabel.HorizontalAlignment = 'center';
            app.N0_ViewModeDropDownLabel.Tooltip = {''};
            app.N0_ViewModeDropDownLabel.Layout.Row = 1;
            app.N0_ViewModeDropDownLabel.Layout.Column = 16;
            app.N0_ViewModeDropDownLabel.Text = 'View Mode';


            % Create N0_ViewModeDropDown
            app.N0_ViewModeDropDown = uidropdown(app.N0_GridLayout);
            app.N0_ViewModeDropDown.Items = {'imagesc', 'imshow'};
            app.N0_ViewModeDropDown.Tooltip = {''};
            app.N0_ViewModeDropDown.Layout.Row = 1;
            app.N0_ViewModeDropDown.Layout.Column = 17;
            app.N0_ViewModeDropDown.Value = 'imagesc';


            % Create N0_TooltipsButton
            app.N0_TooltipsButton = uibutton(app.N0_GridLayout, 'state');
            app.N0_TooltipsButton.Text = 'Tooltips';
            app.N0_TooltipsButton.Layout.Row = 1;
            app.N0_TooltipsButton.Layout.Column = 7;


            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end


    % App creation and deletion
    methods (Access = public)


        % Construct app
        function app = HHGUIApp

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```