What’s New in MATLAB

Joe Hicklin & Ned Gulley

MATLAB EXPO 2019
UNITED KINGDOM

2 October | Silverstone, Northamptonshire
Release Notes for MATLAB - MATLAB & Simulink - MathWorks
https://www.mathworks.com/help/matlab/release-notes

MATLAB Release Notes

R2019b

Environment
- Live Editor Tasks: Add interactive tasks to live scripts to explore parameters and automatically generate code
- Live Editor Output Animates plots to show changes in data over time
- Live Editor Output Adjust the width of columns in tables
- Live Editor Output Scroll through and copy data in arrays such as cell arrays, object arrays, and struct arrays
- Live Editor Export Customizes figure format as well as document paper size, orientation, and margins when exporting
- Live Editor Code: Duplicate one or more lines of code
- Live Editor Code: Suppress Code Analyzer warning messages
- Live Editor Debugging: Set breakpoints for anonymous functions
- Live Editor Internationalization: Add non-English language such as Chinese, Japanese, and Korean characters on Windows and macOS Platforms
- Add-On Manager: Update MATLAB and other installed add-ons
- Add-On Manager: Programmatically manage add-ons by name
- Settings: Create persistent settings for custom apps, toolboxes, and across MATLAB sessions
- MATLAB Drive: Share folders and collaborate with others
- Functionality being removed or changed

Language and Programming
- size Function: Find lengths of multiple array dimensions at a time
- matches Function: Determine if input strings are equal
- Hexadecimal and Binary Numbers: Specify numbers using hexadecimal and binary literals
- Indexing: Use dot indexing into function calls

Compatibility Considerations
MATLAB Release Notes

Bug Reports  Bug Fixes

R2019b

Release Range: R2019a to R2019b

Sort by: Release: Latest to Earliest

New Features, Bug Fixes, Compatibility Considerations

Environment

- Live Editor Tasks: Add interactive tasks to live scripts to explore parameters and automatically generate code
- Live Editor Output: Animate plots to show changes in data over time
- Live Editor Output: Adjust the width of columns in tables
- Live Editor Output: Scroll through and copy data in arrays such as cell arrays, object arrays, and struct arrays
- Live Editor Export: Customize figure format as well as document paper size, orientation, and margins when exporting
- Live Editor Code: Duplicate one or more lines of code
- Live Editor Code: Suppress Code Analyzer warning messages
- Live Editor Debugging: Set breakpoints for anonymous functions
- Live Editor Internationalization: Add non-English language such as Chinese, Japanese, and Korean characters on Windows and macOS Platforms

Add-On Manager: Update MATLAB and other installed add-ons

Contents

- Release Notes
- MATLAB
  - Environment
  - Language and Programming
  - Data Analysis
  - Data Import and Export
  - Mathematics
  - Graph
  - App Building
  - Performance
  - Software Development Tools
  - External Language Interfaces
  - Hardware Support
- Simulink
- 5G Toolbox
- Aerospace Blockset
- Aerospace Toolbox
MATLAB Release Notes

Bug Reports  Bug Fixes

Found 208 notes  Release Range: R2019a to R2019b

Environment
- Live Editor Tasks: Add interactive tasks to live scripts to explore parameters and automatically generate code
- Live Editor Output: Animate plots to show changes in data over time
- Live Editor Output: Adjust the width of columns in tables
- Live Editor Output: Scroll through and copy data in arrays such as cell arrays, object arrays, and struct arrays
- Live Editor Export: Customizes figure format as well as document paper size, orientation, and margins when exporting
- Live Editor Code: Duplicate one or more lines of code
- Live Editor Code: Suppress Code Analyzer warning messages
- Live Editor Debugging: Set breakpoints for anonymous functions
- Live Editor Internationalization: Add non-English language such as Chinese, Japanese, and Korean characters on Windows and macOS Platforms
- Add-On Manager: Update MATLAB and other installed add-ons
- Add-On Manager: Programmatically manage add-ons by name
- Settings: Create persistent settings for custom apps, toolboxes, and across MATLAB sessions

Language and Programming
- size Function: Find lengths of multiple array dimensions at a time
- isclass Function: Determine if input strings are equal
- Hexadecimal and Binary Numbers: Specify numbers using hexadecimal and binary literals
- Indexing: Use dot indexing into function calls
Software Development Tools
- checkcode Function: Get the modified cyclomatic complexity of functions
- Source Control Integration: Synchronize MATLAB Git status with external Git clients
- Unit Testing Framework: Display code coverage metrics in HTML format
- Unit Testing Framework: Specify sources for collections of code coverage data with runtests
- Unit Testing Framework: runperf collects more samples to achieve its target margin of error
- Unit Testing Framework: Return performance test results as Timelapse arrays
- Unit Testing Framework: Load previously saved MeasurementResult objects as DefaultMeasurementResult
- Unit Testing Framework: Use mcat, unittest, fixtures, fixture.onFailure method only in subclasses
- Unit Testing Framework: Compare tables that contain no rows
- Unit Testing Framework: Create test suite array from tests in project
- Unit Testing Framework: Run tests from files in project using runtests or testsuite
- Unit Testing Framework: Specify verbosity, enumeration as a string or character vector
- App Testing Framework: Perform hover gesture on axes, UI axes, and UI figures
- App Testing Framework: Perform press gesture on axes, UI axes, and UI figures
- App Testing Framework: Perform type gesture on date picker objects
- Mocking Framework: Create mocks for classes that use custom maticlasses
- Mocking Framework: Create mocks for classes that use property validation
- Mocking Framework: Specify which methods to mock
- Functionality being removed or changed

External Language Interfaces
- C++: Use C++ classes from third-party libraries in MATLAB
- Python: Version 3.7 support
- Python: Data type support
- C++ MEX: Execute MEX function out of process
- MEX functions: Use customer version of Boost library
- MATLAB Data Array: Support for row-major memory layout
- Compiler support changed for building MEX files and standalone MATLAB engine and MAT-file applications

Hardware Support
- MATLAB Support Package for Parrot Drones: Control Parrot Mambo FPV drone from MATLAB and acquire sensor data
- Deploy Sensei HAT functions on Raspberry Pi hardware
- Functionality being changed or removed
1. The Unit Testing Framework now returns performance test results as arrays of TimeResult objects.
In which our story begins
[The Globe Theatre, Bankside.]
What’s New in MATLAB (the really good bits)
Cast (in order of appearance)

- Projects
- Git Integration
- Live Tasks
- Function Argument Validation
- Live Controls
- AppDesigner Adaptive Layout
- Web Apps
Scene I.

Hot Stuff
Thermostats, Ltd
SpatCo
Thermostat Data (expected)

\[ t_{\text{actual}} = 100 \times 1 \text{ timetable} \]

<table>
<thead>
<tr>
<th>Time</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 hr</td>
</tr>
<tr>
<td>2</td>
<td>0.1 hr</td>
</tr>
<tr>
<td>3</td>
<td>0.2 hr</td>
</tr>
<tr>
<td>4</td>
<td>0.3 hr</td>
</tr>
<tr>
<td>5</td>
<td>0.4 hr</td>
</tr>
<tr>
<td>6</td>
<td>0.5 hr</td>
</tr>
<tr>
<td>7</td>
<td>0.6 hr</td>
</tr>
<tr>
<td>8</td>
<td>0.7 hr</td>
</tr>
<tr>
<td>9</td>
<td>0.8 hr</td>
</tr>
<tr>
<td>10</td>
<td>0.9 hr</td>
</tr>
</tbody>
</table>

![Thermostat Data Graph](image)
Thermostat Data (actual)
In which Ned learns about Projects
Hi Joe:

Run the attached file.

    plot_therm_data.mlx

Ned.
SpatCo Thermostats

This data is a mess. What can I do?

1. load therm_data.mat
2. plot_therm_data(t_cmd, t_measured)
SpatCo Thermostats

This data is a mess. What can I do?

```matlab
load therm_data.mat

Error using load
Unable to read file 'therm_data.mat'. No such file.

plot_therm_data(t_cmd, t_measured)
```
t = simout.yout{1}.Values.Time;
y = simout.yout{1}.Values.Data;

% Make timetables
    t_measured = timetable(hours(t),y);
    t_measured = retime(t_measured,newTime,'linear');
    t_cmd = retime(t_cmd,newTime,'linear');

    plot_therm_data(t_cmd, t_measured);

unrecognized function or variable
plot_therm_data'.
% Make timetables

```
t_measurement = timetable(hours(t), y);
t_measurement = retime(t_measurement, newTime, 'linear');
t_cmd = retime(t_cmd, newTime, 'linear');

plot_therm_data(t_cmd, t_measurement);
```

error: invalid use of plot function.

```
sim_script>plot_therm_data (line 47)
plot(t_measurement.Time, t_measurement.y, 'Marker', 'x',
```
MATLAB Projects
MATLAB Projects
MATLAB Projects
MATLAB Projects
### MATLAB Projects

![MATLAB Projects Screenshot](image)

<table>
<thead>
<tr>
<th>File Name</th>
<th>Git</th>
<th>Size</th>
<th>Date Modified</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slpj</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spatco_app</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get_therm_data.m</td>
<td></td>
<td>2 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>hqi_square.jpg</td>
<td></td>
<td>55 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>makeColorMap.m</td>
<td></td>
<td>1 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>officeSelected.m</td>
<td></td>
<td>1 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>plant.mel</td>
<td></td>
<td>2 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>plot_spatco_floorplan.m</td>
<td></td>
<td>3 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>plot_therm_data.m</td>
<td></td>
<td>1 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>README.md</td>
<td></td>
<td>1 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>sim_script.mlx</td>
<td></td>
<td>224 KB</td>
<td>9/19/2019 3:37 PM</td>
<td>✓</td>
</tr>
<tr>
<td>spatco.mix</td>
<td></td>
<td>720 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>spatco.png</td>
<td></td>
<td>498 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>spatco_app.mapp</td>
<td></td>
<td>195 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>spatco_app.png</td>
<td></td>
<td>3 KB</td>
<td>9/18/2019 10:48 PM</td>
<td>✓</td>
</tr>
<tr>
<td>spatco_office_picker.m</td>
<td></td>
<td>3 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>therm_19b.slx</td>
<td></td>
<td>22 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>therm_19b_svgc</td>
<td></td>
<td>6 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
<tr>
<td>ToolboxPackagingConfiguration.m</td>
<td></td>
<td>7 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✓</td>
</tr>
</tbody>
</table>
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.
```

```matlab
plot_therm_data(t_cmd, t_actual)
```

Thermostat Data

- [Blue line] Measured
- [Red line] Desired
This is a sample MATLAB code snippet. It appears to be a function that reads thermal data and plots a graph.

```matlab
[t_therm, m_therm] = get_therm_data(sys);
```

The comments indicate that the data might be expected to look like this in a zero-noise situation.
Hi Joe:

Here’s the project URL.

https://github.com/HotStuff/SpatCo

Ned.
In which Joe receives a Project from Ned
To perform branch merges, you must have command-line Git installed.
If you do not have command-line Git installed already, follow the instructions here.
Source control tool: Git
Repository path: https://insidelabs-git.mathworks.com/gulley/uk2019
Sandbox: C:\Users\joe\MATLAB\Projects\untitled
Source control information: To perform branch merges, you must have command-line Git installed. If you do not have command-line Git installed already, follow the instructions here.
To perform branch merges, you must have command-line Git installed. If you do not have command-line Git installed already, follow the instructions here.
To perform branch merges, you must have command-line Git installed. If you do not have command-line Git installed already, follow the instructions below.
Retrieving files from repository:

Receiving objects

Stop
UK 2019
Root: \L\Dropbox\UK talk\Needs Study
Analyzed: 9/17/2019 2:46 PM

Products (3)
- MATLAB 9.7
- Simulink 10.0
- Simulink Coder 9.2
SpatCo Thermostats

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

```
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.
In which Joe uses Live Tasks
Thermostat Data (actual)
Thermostat Data (actual)
Thermostat Data (actual)
Live Tasks
Live Tasks

Find Local Extrema

maxIndices = Local maxima in PeakSig

Select data
Input data: PeakSig X-axis: default

Define local extrema
Extrema type: Maxima Flat selection: Center
Max. num. extrema: 1000 Min. prominence: 0
Min. separation: 0 Prominence window: Centered Centered 100

Visualize results
Local maxima: checked Input data: checked
Live Tasks

```matlab
% Find local maxima
maxIndices = islocalmax(PeakSig);

% Visualize results
clf
plot(PeakSig,'Color',[109 185 226]/255,'DisplayName','Input data')
hold on

% Plot local maxima
plot(find(maxIndices),PeakSig(maxIndices),'^','Color',[217 83 25]/255,...
     'MarkerFaceColor',[217 83 25]/255,'DisplayName','Local maxima')
title([Number of extrema: num2str(nnz(maxIndices))]')
hold off
legend
```
% Set up the simulation

tFinal = 100;

smin = simulink.open('simulink19b');
sim = smin.simulink('Start', '0', 'StopTime', num2str(tFinal));

% smin = smin.setModelParameters('simulink19b', 'StateSpace', 'sys.A')

% Pick some random times to change the commanded temperature

t = sort(tFinal * rand(10, 1));
t(1) = 0;
cmd = rand(20, size(t(1))) + 10;

% cmd = repmat(hour(t(1), cmd);

dt = 0.1;
newTime = hours(0:dt:tFinal);

t_cmd = settime(t_cmd, newTime, 'previous');

% Run the simulation

smin = smin.setExternalInput([hours(t_cmd, t_cmd, cmd]);

simout = sim(smin);

% Find local maxima

maxIndices = islocalmax(PeakSig);

% Visualize results

clf
plot(PeakSig, 'Color', [180 38 226]/255, 'DisplayName', 'Input data')
hold on

% Plot local maxima

plot(find(maxIndices), PeakSig(maxIndices), 'r', 'Color', [217 83 25]/255,

'tooltip', 'true', 'DisplayName', 'Local maxima')

title('Number of extrema: ' num2str(nmax(maxIndices)));
hold off

legend

ybad = t_actual.y;

% noise

ybad = ybad + 3*randn(size(ybad));

% drop-outs

numbad = rand(size(ybad)) > 0.8;
ybad(numbad) = 0;

% missing

numbad = rand(size(ybad)) > 0.4;
ybad(numbad) = NaN;

yMeasured.y = ybad;
% Set up the simulation

tFinal = 100;

simin = Simulink.EnvironmentInput('therm_19b');
simin = simin.setSimulationParameter('StartTime',0,'StopTime',num2str(tFinal));

% simin = simin.setSimulationParameter('therm_19b\StateSpace',A,sigma.A)

% Pick some random times to change the commanded temperature

t = sort(tFinal*randn(10,1));
t(1) = 0;
cmd = randi([20,25],[20,1]) + 10;
t_cmd = timetable(hours(t),cmd);

dt = 0.1;
newTime = hours(0:dt:tFinal);

t_cmd = settime(t_cmd,newTime,'previous');

% Run the simulation

simin = simin.setExternalInput([hours(t_cmd.Time), t_cmd.cmd]);
simout = sim(simin);

Find Local Extrema

Select data
input data: [ 1 2 3 4 5 6 7 8 9 10 ]

Define local extrema

Visualize results

ybad = t_actual.y;

% noise
ybad = ybad + 3*randn(size(ybad));

% drop-outs
imbad = rand(size(ybad)) > 0.5;
ybad(imbad) = NaN;

% missing
imbad = rand(size(ybad)) > 0.4;
ybad(imbad) = NaN;

t_measured.y = ybad;

Function [t_cmd, t_measured, t_actual] = get_therm_data(tyblock)
SpatCo Thermostats

1. \( \{t_{cmd}, t_{measured}, t_{actual}\} = \text{get\_therm\_data}(sys); \)
   This is what it should look like in a zero-noise situation. I expect the data to look like this.
2. \( \text{plot\_therm\_data}(t_{cmd}, t_{actual}) \)
   But this is what it actually looks like. Instead, it’s a mess, as shown below.
3. \( \text{plot\_therm\_data}(t_{cmd}, t_{measured}) \)
   I need help trying to recapture what true temperature profile is.
SpatCo Thermostats

```plaintext
[t_cmd, t_measured, t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.

plot_therm_data(t_cmd, t_actual)

But this is what it actually looks like. Instead, it's a mess, as shown below.

plot_therm_data(t_cmd, t_measured)

I need help trying to recapture what true temperature profile is.
```
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to be clean.

```matlab
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```matlab
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.
SpatCo Thermostats

1. `[t_cmd, t_measured, t_actual] = get_therm_data(sys);`  
   This is what it should look like in a zero-noise situation. I expect the data to look like this.

2. `plot_therm_data(t_cmd, t_actual)`  
   But this is what it actually looks like. Instead, it's a mess, as shown below.

3. `plot_therm_data(t_cmd, t_measured)`  
   I need help trying to recapture what true temperature profile is.

---

Clean Outlier Data

Find, fill, or remove outliers

- Select data
  - Input data: select ▼ X-axis: default ▼
- Specify cleaning method
  - Cleaning method: ▼
  - Define outliers
    - Detection method: Median ▼ Threshold factor: 3 ▼
- Visualize results
  - □ Filled outliers □ Cleared data □ Input data □ Outliers □ Outlier thresholds □ Outlier center

---

Thermostat Data

- **Measured**
- **Closed**
SpatCo Thermostats

1. \([t_{\text{cmd}}, t_{\text{measured}}, t_{\text{actual}}]\) = get_therm_data(sys);

   This is what it should look like in a zero-noise situation. I expect the data to look like this.

   plot_therm_data(t_{\text{cmd}}, t_{\text{actual}})

   But this is what it actually looks like. Instead, it looks like the one shown below:

   plot_therm_data(t_{\text{cmd}}, t_{\text{measured}})

   I need help trying to recapture what the thermostat profile is.

---

Clean Outlier Data

- Find, fill, or remove outliers

Select data
- Input data: select
- X-axis: default

Specify cleaning method
- Cleaning method: Fill outliers

Define detection
- Detection method: Median
- Threshold factor: 3

Visualize results
- Fill outliers
- Cleaned data
- Input data
- Outliers
- Outlier thresholds
- Outlier center

Thermostat Data

- Measured
- Cleaned
SpatCo Thermostats

```
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

---

Clean Outlier Data

Find, fill, or remove outliers

**Select data**
- Input data: select

**Specify**
- X-axis: default

**Define outliers**
- Detection method: Median
- Threshold factor: 3

**Visualize results**
- Filled outliers
- Cleaned data
- Input data
- Outliers
- Outlier thresholds
- Outlier center

---

Thermostat Data

```
Measured
Closed
```

---

Thermostat Data

```
Temperature vs. Time
```

---

Thermostat Data
SpatCo Thermostats

```
[t_cmd; t_measured; t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

---

**Clean Outlier Data**

**(cleanedData)** - Filled outliers in `t_measured` using the linear interpolation method

**Select data**

- **Input data**: `t_measured`
- **y**: `t_measured`
- **X-axis**: `t_measured`
- **Time**

**Specify cleaning method**

- **Cleaning method**: Fill outliers
- **Linear interpolation**

**Define outliers**

- **Detection method**: Median
- **Threshold factor**: 3

**Visualize results**

- **Filled outliers**
- **Cleaned data**
- **Input data**
- **Outliers**
- **Outlier thresholds**
- **Outlier center**

---

**Thermostat Data**

- **Input data**
- **Cleaned data**
- **Outliers**
- **Filled as aliases**
- **Outlier thresholds**
- **Outlier center**

**Number of outliers**: 79

---

**Temperature vs. Time**

- **Time (y-axis)**
- **Temperature (x-axis)**

---

**Number of outliers**: 79
SpatCo Thermostats

```matlab
[t_cmd,t_measured,t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.

plot_therm_data(t_cmd, t_actual)

But this is what it actually looks like. Instead, it’s a mess, as shown below.

plot_therm_data(t_cmd, t_measured)

I need help trying to recapture what true temperature profile is.
```

Clean Outlier Data

- **Input data**: `t_measured` for `y` using the linear interpolation method.
- **Select data**: `t_measured` for `y` and `Time` for `X-axis`.
- **Specify cleaning method**: Fill outliers → Linear Interpolation.
- **Define outliers**: Median → Threshold factor 3.
- **Visualize results**: Fill outliers, Cleaned data, Input data, Outliers, Outlier thresholds, Outlier center.
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```
This is what it should look like in a zero-noise situation. I expect the data to look like this.

```matlab
plot_therm_data(t_cmd, t_actual)
```
But this is what it actually looks like. Instead, it's a mess, as shown below:

```matlab
plot_therm_data(t_cmd, t_measured)
```
I need help trying to recapture what true temperature profile is.

**Clean Outlier Data**

(cleanedData) = Clean outliers in t_measured_y using the linear interpolation method

**Select data**

- Input data: t_measured, y, X-axis, t_actual, Time

**Specify cleaning method**

- Cleaning method: Fill outliers, Linear Interpolation

**Define outliers**

- Detection method: Median, Threshold factor: 3

**Visualize resul**

- Median, Mean
- Visualize outliers: Quartiles, Grubbs
- Generalized extreme studentized deviate (GESD)

Number of outliers: 79

**Thermostat Data**

- Measured
- Cleaned

Temperature over time

- Time range: 0 to 100
- Number of outliers: 79

Input data: Measured, Cleaned data, Outliers

- Filled as missing
- Outlier threshold, Outlier center
SpatCo Thermostats

```
[t_cmd, t_measured, t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.

plot_therm_data(t_cmd, t_actual)

But this is what it actually looks like. Instead, it's a mess, as shown below:

plot_therm_data(t_cmd, t_measured)

I need help trying to recapture what true temperature profile is.
```

Clean Outlier Data

(cleanedData) = Clean outliers in t_measured_y using the linear interpolation

Select data

Input data: t_measured y X-axis: t_measured Time

Specify cleaning method

Cleaning method: Fill-outliers Linear Interpolation

Define outliers

Detection method: Moving median Threshold factor: 2.5

Moving window: Centered Units: Days

Visualize results

Filled outliers Cleaned data Input data Outliers Outlier thresholds

Number of outliers: 79

Thermostat Data

```

Temperature (K)/Time (s) for Cleaned and Measured Thermostat Data

Number of outliers: 79
```
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
This is what it should look like in a zero-noise situation. I expect the data to look like this.
plot_therm_data(t_cmd, t_actual)
But this is what it actually looks like. Instead, it's a mess, as shown below:
plot_therm_data(t_cmd, t_measured)
I need help trying to recapture what true temperature profile is.

Clean Outlier Data
(cleanedData) = Fill outliers in t_measured_y using the linear interpolation method

Select data
Input data: t_measured_y
X-axis: t_measured, Time

Specify cleaning method
Cleaning method: Fill outliers, Linear interpolation

Define outliers
Detection method: Moving median, Threshold factor 2.5
Moving window: Centered, 1, Units Days

Visualize results
Filled outliers, Cleaned data, Input data, Outliers, Outlier center
```

Thermostat Data

Number of outliers: 79
Clean Outlier Data

(cleanedData) = Filled outliers in t measured, y using the linear interpolation method

Select data
Input data: measured, y, x
X-axis: t measured, time

Specify cleaning method
Cleaning method: Fill outliers, Linear interpolation

Define outliers
Detection method: Moving median, Threshold factor
Moving window: Centered, 1

Visualize results
☑ Filled outliers, Cleaned data
☑ Input data, Outliers, Outlier thresholds
☑ Outlier center

```matlab
% fill outliers
(cleanedData, outlierIndices, thresholdLow, thresholdHigh) = ... filloutliers(t measured, y, 'linear', 'movingMedian', 'days', 1,... 'ThresholdFactor', 2.5, 'SamplePoints', t measured, time);
% visualize results
cf
plot(t measured, t measured, 'Color', [1 0.7 0 1/255], DisplayName, 'Input data')
hold on
plot(t measured, cleanedData, 'Color', [0 1 0 1/255], 'LineWidth', 1.5,... DisplayName, 'Cleaned data')
% plot outliers
plot(t measured, outlierIndices, 'Color', [1 0 0 1/255], DisplayName, 'Outliers')
title(['Number of outliers: ' num2str(size(outlierIndices))])
% plot filled outliers
plot(t measured, cleanedData2(outlierIndices), 'Color', [1 0.4 0 1/255], DisplayName, 'Filled outliers')
% plot threshold high
plot(t measured, missing, t measured, 'Color', [1 0.4 0 1/255], DisplayName, 'Threshold high')
```
Clean Outlier Data
(cleanedData) = Fill outliers in t_measured,y using the linear interpolation method

Select data
Input data: t_measured, y
X-axis: t_measured, Time

Specify cleaning method
Cleaning method: Fill outliers, Linear interpolation

Define outliers
Detection method: Moving median, Threshold factor: 2.5
Moving window: Centered, 1, 2, Units: Days

Visualize results
Filled outliers, Cleaned data, Input data, Outliers, Outlier thresholds, Outlier center

% fill outliers
[cleanedData, outlierIndices, thresholdLow, thresholdHigh] = ...
    filloutliers(t_measured,y,", linear", ',median',days,1)
    'ThresholdFactor',2.5, 'SamplePoints',t_measured.Time);

% Visualize results
clf
plot(t_measured.Time,t_measured.y,'Color',[109 185 220]/255,...
    'DisplayName', 'Input data')
hold on
plot(t_measured.Time,cleanedData,'Color',[0 114 189]/255,'LineWidth',1.5,...
    'DisplayName', 'Cleaned data')

% Plot outliers
plot(t_measured.Time(outlierIndices),t_measured.y(outlierIndices),',k',...
    'Color',[64 04 64]/255, 'DisplayName', 'Outliers')
title(['Number of outliers: ', num2str(sum(outlierIndices))])

% Plot filled outliers
plot(t_measured.Time(outlierIndices),cleanedData2(outlierIndices),',r',...
    'MarkerSize',12, 'Color',[217 83 25]/255, 'DisplayName', 'Filled outliers')

% Plot outlier thresholds
plot([thresholdHigh; missing; thresholdLow; thresholdLow; missing],...
    [145 146 141]/255,...
Clean Outlier Data
(cleanedData) = Filled outliers in \( t_{\text{measured}}, y \) using the linear interpolation method.

Select data
Input data \( t_{\text{measured}}, y \), X-axis \( t_{\text{measured}}, \text{Time} \).

Specify cleaning method
Cleaning method: Fill outliers, Linear interpolation.

Define outliers
Detection method: Moving median, Threshold factor 2.5.
Moving window: Centered, 1, Units: Days.

Visualize results
- Display cleaned data
- Display input data
- Display outliers
- Display threshold values

```
% Visualize results
clf
plot(t_measured.Time, t_measured.y, [1 0.1 0.6 0.5]/255, 'DisplayName', 'Input data');
hold on
plot(t_measured.Time, cleanedData, [0.1 0.1 0.8 0.8]/255, 'LineWidth', 1.5, 'DisplayName', 'Cleaned data');

% Plot outliers
plot(t_measured.Time(outlierIndices), t_measured.y(outlierIndices), [0.6 0.4 0.6]/255, 'DisplayName', 'Outliers');
title([Number of outliers: ' num2str(numoutliers(outlierIndices))]);

% Plot filled outliers
plot(t_measured.Time(outlierIndices), cleanedData2(outlierIndices), [0.4 0.2 0.8]/255, 'DisplayName', 'Filled outliers');

% Plot threshold values
plot(t_measured.Time, missing, t_measured.Time, [0.2 0.4 0.8]/255, 'DisplayName', 'Filled values');
```
Clean Outlier Data

CleanedData

Select data
Input data  \( t_{\text{measured}}, y \)  \( X \)-axis \( t_{\text{measured}}, y \)  Time

Specify cleaning method
Cleaning method  Fill outliers  Linear interpolation

Define outliers
Detection method  Moving median  Threshold factor 2.5
Moving window  Centered

Visualize results
\( \check{\text{Fill outliers}} \)  \( \check{\text{Cleaned data}} \)  \( \check{\text{Input data}} \)  \( \check{\text{Outliers}} \)  \( \check{\text{Outlier thresholds}} \)  \( \check{\text{Outlier center}} \)

```
% fill outliers
[cleanedData, outlierIndices, thresholdLow, thresholdHigh] = filloutliers(t_measured, y, 'linear', 'movingmedian', days(1),
    'ThresholdFactor', 2.5, 'SamplePoints', t_measured.Time);

% Visualize results
clf
plot(t_measured.Time, t_measured, 'Color', [109 185 202] / 255,
    'DisplayName', 'Input data');
hold on
plot(t_measured.Time, cleanedData, 'Color', [0 114 180] / 255, 'LineWidth', 1.5,
    'DisplayName', 'Cleaned data');

% Plot outliers
plot(t_measured.Time(outlierIndices), t_measured, 'Color', [64 04 64] / 255,
    'DisplayName', 'Outliers');
title(['Number of outliers: ' num2str(size(outlierIndices))]);

% Plot filled outliers
plot(t_measured.Time(outlierIndices), cleanedData2, 'Color', [217 83 25] / 255,
    'DisplayName', 'Filled outliers');

% Plot outlier thresholds
plot(t_measured.Time, thresholdLow, 'Color', [145 145 141] / 255,
    'DisplayName', 'Lower threshold');
plot(t_measured.Time, thresholdHigh, 'Color', [148 148 148] / 255,
    'DisplayName', 'Upper threshold');
```
SpatCo Thermostats

[t_cmd, t_measured, t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.

plot_therm_data(t_cmd, t_actual)

But this is what it actually looks like. Instead, it's a mess, as shown below.

plot_therm_data(t_cmd, t_measured)

I need help trying to recapture what true temperature profile is.

cleanedData2 = Filled outliers in t_measured_y using the linear interpolation method
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);

This is what it should look like in a zero-noise situation. I expect the data to look like this.

plot_therm_data(t_cmd, t_actual)

But this is what it actually looks like. Instead, it's a mess, as shown below:

plot_therm_data(t_cmd, t_measured)

I need help trying to recapture what true temperature profile is.

-cleanedData2 = Filled outliers in t_measured, y using the linear interpolation method

m15
```

- missing
- missing
- missing
- missing
- misdata
- Clean Missing Data
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```
This is what it should look like in a zero-noise situation. I expect the data to look like this.

```matlab
plot_therm_data(t_cmd, t_actual)
```
But this is what it actually looks like. Instead, it's a mess, as shown below.

**mis**

- **missing**: Create missing values
- **mislocked**: Determine if function or script is locked in memory
- **misdata**: Reconstruct missing input and output data
- **Clean Missing Data**: Find, fill, or remove missing data
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```matlab
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```matlab
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

- `cleanedData` - Filled outliers in `t_measured` using the linear interpolation method

### Clean Missing Data

Find, fill, or remove missing data

- **Select data**
  - Input data
  - X-axis
- **Specify method**
  - Cleaning method: Fill missing
- **Visualize results**
  - Cleaned data
  - Filled missing entries

Number of outliers: 79
SpatCo Thermostats

```
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below:

```
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

- **cleanedData** → Filled outliers in t_measured.y using the linear interpolation method

### Clean Missing Data

- **cleanedData** → Filled missing data in cleanedData using the linear interpolation method

**Select data**

- **Input data**: cleanedData
- **X-axis**: default

**Specify method**

- **Cleaning method**: Fill missing
- **Linear interpolation**:

**Visualize results**

- **Cleaned data**
- **Filled missing entries**

---

Number of outliers: 79

Number of filled missing entries: 583
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```matlab
plot_therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```matlab
plot_therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

- cleanedData - Filled outliers in t_measured with linear interpolation method
- cleanedData2 - Filled missing data in cleanedData using the linear interpolation method

### Smooth Data

- smoothedData - Smoothed noisy data in cleanedData2 using the moving mean method

**Select data**

- Input data: cleanedData2
- X-axis: default

**Specify method and parameters**

- Smoothing method: Moving mean
  - Smoothing factor: 0.38

**Visualize results**

- Smoothed data
- Input data
SpatCo Thermostats

```
[t_cmd, t_measured, t_actual] = get_therm_data(sys);
This is what it should look like in a zero-noise situation. I expect the data to look like this.
plot_therm_data(t_cmd, t_actual)
But this is what it actually looks like. Instead, it's a mess, as shown below.
plot_therm_data(t_cmd, t_measured)
I need help trying to recapture what true temperature profile is.
```

- cleanedData = Filled outliers in t_measured using the linear interpolation method
- cleanedData2 = Filled missing data in cleanedData using the linear interpolation method
- smoothedData = Smoothed noisy data in cleanedData2 using the moving mean method
SpatCo Thermostats

```matlab
[t_cmd, t_measured, t_actual] = get Therm_data(sys);
```

This is what it should look like in a zero-noise situation. I expect the data to look like this.

```matlab
plot Therm_data(t_cmd, t_actual)
```

But this is what it actually looks like. Instead, it's a mess, as shown below.

```matlab
plot Therm_data(t_cmd, t_measured)
```

I need help trying to recapture what true temperature profile is.

Ned, I've added these three data cleaning steps. Why don't you take a look at them and see if you think they are appropriate?

- cleanedData - Filled outliers in t_measured_y using the linear interpolation method
- cleanedData2 - Filled missing data in cleanedData using the linear interpolation method
- smoothedData - Smoothed noisy data in cleanedData2 using the moving mean method
In which Joe validates function arguments
Argument Validation

```plaintext
function showTempData( fileName, maxDataPoints )

% blah
% blah
```
function showTempData( fileName, maxDataPoints )

arguments
    fileName (1,1) string
    maxDataPoints (1,1) double { mustBePositive } = 10000
end

% blah
% blah
Argument Validation

```
function showTempData(filename, maxDataPoints)
    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
Argument Validation

```matlab
function showTempData( fileName, maxDataPoints )

    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
Argument Validation

```plaintext
function showTempData( fileName, maxDataPoints )

    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
Argument Validation

```matlab
function showTempData( fileName, maxDataPoints )

    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
Argument Validation

```plaintext
function showTempData( fileName, maxDataPoints )
  arguments
    fileName (1,1) string
    maxDataPoints (1,1) double { mustBePositive } = 10000
  end

% blah
% blah
```
Argument Validation

```matlab
function showTempData( fileName, maxDataPoints )
    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
function showTempData( fileName, maxDataPoints )

arguments
    fileName (1,1) string
    maxDataPoints (1,1) double
    {
        mustBePositive
    } = 10000
end

% blah
% blah
MATLAB Validation Functions

MATLAB defines functions for use in property validation. These functions support common use patterns for validation and provide descriptive error messages. This table lists the MATLAB validation functions, their meanings, and the MATLAB functions used by the validation functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Functions Called on Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>mustBePositive(value)</td>
<td>value &gt; 0</td>
<td>gt, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeNonpositive(value)</td>
<td>value &lt;= 0</td>
<td>ge, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeFinite(value)</td>
<td>value has no NaN and no Inf elements.</td>
<td>infinite</td>
</tr>
<tr>
<td>mustBeNonNan(value)</td>
<td>value has no NaN elements.</td>
<td>isnan</td>
</tr>
<tr>
<td>mustBeNonnegative(value)</td>
<td>value &gt;= 0</td>
<td>ge, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeNegative(value)</td>
<td>value &lt; 0</td>
<td>lt, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeNonzero(value)</td>
<td>value ~= 0</td>
<td>eq, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeGreaterThan(value,c)</td>
<td>value &gt; c</td>
<td>gt, isscalar, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeLessThan(value,c)</td>
<td>value &lt; c</td>
<td>lt, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeGreaterThanOrEqual(value,c)</td>
<td>value &gt;= c</td>
<td>ge, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeLessThanOrEqual(value,c)</td>
<td>value &lt;= c</td>
<td>le, isreal, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeNonempty(value)</td>
<td>value is not empty</td>
<td>isempty</td>
</tr>
<tr>
<td>mustBeNonsparse(value)</td>
<td>value has no sparse elements.</td>
<td>issparse</td>
</tr>
<tr>
<td>mustBeNumeric(value)</td>
<td>value is numeric</td>
<td>isnumeric</td>
</tr>
<tr>
<td>mustBeNumericOrLogical(value)</td>
<td>value is numeric or logical.</td>
<td>isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeReal(value)</td>
<td>value has no imaginary part.</td>
<td>isreal</td>
</tr>
<tr>
<td>mustBeInteger(value)</td>
<td>value == floor(value)</td>
<td>isreal, isnfinite, floor, isnumeric, islogical</td>
</tr>
<tr>
<td>mustBeMember(value,S)</td>
<td>value is an exact match for a member of S.</td>
<td>ismember</td>
</tr>
</tbody>
</table>
Argument Validation

```matlab
function showTempData( fileName, maxDataPoints )

    arguments
        fileName (1,1) string
        maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
Argument Validation

```plaintext
function showTempData( fileName, maxDataPoints )

    arguments
    fileName (1,1) string
    maxDataPoints (1,1) double { mustBePositive } = 10000
    end

    % blah
    % blah
```
In which Ned calls a function
SpatCo Headquarters Data

showTempData
SpatCo Headquarters Data

showTempData

Error using showTempData
Invalid input argument list. Not enough input arguments.
Function requires 1 input(s).
SpatCo Headquarters Data

showTempData()

showTempData(filename, maxDataPoints)

Enter a value for filename
SpatCo Headquarters Data

```python
showTempData("spatco_hq.csv",)
```

```python
showTempData(filename, maxDataPoints)
```

Enter a value for `maxDataPoints` (Optional)
SpatCo Headquarters Data

```
showTempData("spatco_hq.csv",-1)
```
SpatCo Headquarters Data

```
showTempData("spatco_hq.csv",-1)
```

Error using `showTempData`
Invalid input argument at position 2. Value must be positive.
SpatCo Headquarters Data

```
showTempData("spatco_hq.csv", 10000)
```

Loading 10000 data points from spatco_hq.csv
In which Ned uses Live Controls
Specify the floor and visualize

```matlab
floor_index = 4;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```
floor_index = 4;
h = plot_spatco_file
color_offices(h,floor_index,temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 4;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 4;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = ;
h = plot_spatio.floorplan(gca);
color_offices = color_offices(floor_index, room_temp, colors);
```
Specify the floor and visualize

```matlab
floor_index = 9;
h = plot_spatco.floorplan(gca);
color_offices(floor_index, room_temp, colors);
```
Specify the floor and visualize

```python
floor_index = 9;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```plaintext
floor_index = 9;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 9;
h = plot_spatio_floorplan(gca);
color_offices(floor_index, room_temp, colors);
```
Specify the floor and visualize

```matlab
floor_index = 17;
h = plot_spatio_floorplan(gca);
color_offices(floor_index, room_temp, colors);
```
Specify the floor and visualize

```matlab
floor_index = 17;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```
floor_index = 17;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 17;

h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```plaintext
floor_index = 17;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 17;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```
Specify the floor and visualize

```matlab
floor_index = 17;

h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,offices);
```

Floor 17
Specify the floor and visualize

def floor_index = 18;

h = plot_space(
floor_index
);
color_office

LABEL

Text to display when code is hidden

Label: Floor

VALUES

Min: 0
Max: 41
Step: 1

EXECUTION

Run On: Value changed
Run: Current section
Specify the floor and visualize

```matlab
floor_index = Floor 18 ;
h = plot_spare
color_office
```

**LABEL**
Text to display when code is hidden
Label: Floor

**VALUES**
- Min: 0
- Max: 41
- Step: 1

**EXECUTION**
- Run On: Value changed
- Run: Current section
Specify the floor and visualize

```plaintext
floor_index = Floor 18;
h = plot_spans(color_officer);
```

**LABEL**
- Text to display when code is hidden
  - Label: Floor

**VALUES**
- Min: 0
- Max: 41
- Step: 1

**EXECUTION**
- Run On: Value changed
- Run: Current section
Live Controls

- Numeric Slider
- Drop-Down List
  - Male
  - Male
  - Female
- Check Box
- Edit Field
  
Enter text
- Button
  
  Run
Specify the floor and visualize

```matlab
floor_index = 18;
h = plot_spatco_floorplan(gca);
color_offices(h,floor_index,room_temp,colors);
```

Floor 18
Specify the floor and visualize

Floor Number 8

Floor 8
Specify the floor and visualize

Floor Number 13
Specify the floor and visualize

Floor Number 18

Floor 18
Specify the floor and visualize

Floor Number 30
Specify the floor and visualize

Floor Number 18

Floor 18
Specify the floor and visualize

Floor Number 31

Floor 31
In which Ned makes an App with AppDesigner
>> appdesigner
MATLAB® App Designer

New to App Designer? Try a 3-minute tutorial.

- Getting Started
- Apps to App Designer
- Displaying Graphics in App Designer
- Release Notes

**New**
- Blank App
- 2-Panel App with Auto-Reflow
- 3-Panel App with Auto-Reflow

**Examples: General**
- Interactive Tutorial
- Respond to Numerical Input
- Respond to User Selections
- Embed HTML Content
- Layout Controls in a Grid
Create a new 2-panel app that automatically resizes and reflows its layout to accommodate different screen sizes.
SpatCo HQ Thermostats

Floor 6
Office 601

Office Plan for Floor 6
SpatCo HQ
Thermostats

Floor 12
Office 1201

Office Plan for Floor 12
In which Joe creates a Web App.
Web Apps
Web Apps
Web Apps
Web Apps
Web Apps
Web Apps

[Image of a folder tree showing files such as `get_therm_data.m`, `hq-square.jpg`, `makeColorMap.m`, `officeSelected.m`, `plot_spatco_floorplan.m`, `plot_therm_data.m`, `README.md`, `sim_script.mlx`, `spatco.mlx`, `spatco.png`, `spatco_app.mlapp`, `spatco_office_picker.m`, `therm_19b.xlsx`, `therm_19b.xlsx`, `ToolboxPackagingConfiguration.prj`, and `UK2019.prj`.]
Web Apps
Web Apps
Web Apps

Office Heat Explorer version 2.1
by Ned
Thermal Analysis of SpatCo World Headquarters

Building...

Close dialog after packaging
Edit Project
Cancel Packaging
Web Apps

Office Heat Explorer version 2.1
by Ned
Thermal Analysis of SpatCo World Headquarters

Open output folder.

Close dialog after packaging
Edit Project
Close
Web Apps
Web Apps
Web Apps
Web Apps

MATLAB Web Apps

- App 1
- Office Heat Explorer
- Patients Display
- Ugly Data App

Diagnostics

Folder with apps:
- Better Viscosity Calculator.cft
- OfficeHeatExplorer.cft
- Pancake Density Converter.cft
- SpatCo App.cft

Files:
- OfficeHeatExplorer.cft
  - Date modified: 9/11/2019 1:23 PM
  - Type: CTF File
  - Size: 1,107 KB
- Pancake Density Converter.cft
  - Date modified: 9/25/2019 9:17 AM
  - Type: CTF File
  - Size: 2,082 KB
- The Spatulator.cft
  - Date modified: 9/11/2019 2:39 PM
  - Type: CTF File
  - Size: 1,550 KB
- SpatCo App.cft
  - Date modified: 9/11/2019 2:00 PM
  - Type: CTF File
  - Size: 1,167 KB
Hi Ned:

The web app is complete.

Check it out at

http://ah-joe:9988/webapps/home/

Joe.
SpatCo HQ
Thermostats

Floorplan
Thermostat

Office Plan for Floor 1

Floor: 1
Office: 101
One week later...
In which Ned summarizes the Talk
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects

Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Projects
Git Integration
Live Tasks
Function Argument Validation
Live Controls
AppDesigner Adaptive Layout
Web Apps
Thank you!