What’s New in MATLAB

Joe Hicklin & Ned Gulley

MATLAB EXPO 2019
UNITED KINGDOM

2 October | Silverstone, Northamptonshire
MATLAB Release Notes

Found 200 notes | Release Range: R2019a to R2019b

Sort by: Release: Latest to Earliest

New Features, Bug Fixes, Compatibility Considerations

R2019b: Bug Fixes

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
- Add-On Manager: Programatically 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

Compatibility Considerations

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

Check for updates.
MATLAB Release Notes

Bug Reports  Bug Fixes

Found 208 notes  Release Range: R2019a to R2019b

Sort by: Release: Latest to Earliest

Release Range:

R2019a ▼ to ▼ R2019b
MATLAB Release Notes

Bug Reports | Bug Fixes

R2019b

Found 208 notes | Release Range: R2019a to R2019b

Sort by:

Release: Latest to Earliest

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 warnings/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

Compatability Considerations

- Data Analytics
- Data Import and Export
- Mathematics
- Graphics
- App Building
- Performance
- Software Development Tools
- External Language Interfaces
- Hardware Support
- Simulink
- 5G Toolbox
- Aerospace Blockset
- Aerospace Toolbox
- Text Filter

Release Filter:

R2019a to R2019b

Compatibility Considerations

- Incompatibilities Only
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 Timereport arrays
- Unit Testing Framework: Load previously saved measurement objects as DefaultMeasurementResult
- Unit Testing Framework: Use mlelabunitest, 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 mclabcollections
- Mocking Framework: Create mocks for classes that use property validation
- Mocking Framework: Specify which methods to mock

Functionally 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++: Execute MEX function out of process
- MEX functions: Use customer version of Boost library
- MATLAB Data Arrays: 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 Sense HAT functions on Raspberry Pi hardware
- Functionally being changed or removed
The Unit Testing Framework now returns performance test results as arrays of TimeResult objects.

Run performance tests for all the elements that contain 'Indexing' in the name. Your results might vary, and you might see a warning if `runperf` doesn't meet statistical objectives.

```java
results = runperf('preallocationTest','','Indexing')
```

Run `runperf`.

- `results = runperf()` run all the tests in your current folder for performance measurements and returns an array of `matlab.perftest.TimeResult` objects. Each element in `results` corresponds to an element in the test suite.

The performance test framework runs the tests using a variable number of measurements to reach a sample mean with a 0.05 relative margin of error within a 0.95 confidence level. It runs the tests four times to warm up the code, and then between 4 and 256 times to collect measurements that meet the statistical objectives. If the sample mean does not meet the 0.05 relative margin of error within a 0.95 confidence level after 256 test runs, the performance test framework stops running the test and displays a warning. In this case, the `matlab.perftest.TimeResult` object contains information for the 4 warm-up runs and 256 measurement runs.

The `runperf` function provides a simple way to run a collection of tests as a performance experiment.

```java
results = runperf(tests)
results = runperf(tests,Name,Value)
```

Syntax:

- `results = runperf()` runs all the tests in your current folder for performance measurements and returns an array of `matlab.perftest.TimeResult` objects.

- `results = runperf(tests)` runs the specified set of tests.

- `results = runperf(tests,Name,Value)` runs a set of tests with additional options specified by one or more `Name,Value` pair arguments.
In which our story begins
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
SpatCo
Thermostat Data (expected)

t\_actual = 100\times 1\ 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 (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.
```

```matlab
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_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);

Error: Too few input arguments.

In file: sim_script>plot_therm_data (line 47)
plot(t_measured.Time,t_measured.y,'Marker','x',...
MATLAB Projects
MATLAB Projects
MATLAB Projects

<table>
<thead>
<tr>
<th>File</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>hq_square.jpg</td>
<td></td>
<td>55 KB</td>
<td>9/18/2019 10:47 AM</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>9/17/2019 2:46 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.prg</td>
<td></td>
<td>498 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✔️</td>
</tr>
<tr>
<td>spatco_app.m</td>
<td></td>
<td>195 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✔️</td>
</tr>
<tr>
<td>spatco_app.pgp</td>
<td></td>
<td>3 KB</td>
<td>9/18/2019 10:48 AM</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_39b.xlsx</td>
<td></td>
<td>22 KB</td>
<td>9/17/2019 2:46 PM</td>
<td>✔️</td>
</tr>
<tr>
<td>therm_39b.xvc</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>
MATLAB Projects

<table>
<thead>
<tr>
<th>File</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>hq_square.png</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_floormap.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>224KB</td>
<td>9/29/2019 3:37 PM</td>
<td></td>
</tr>
<tr>
<td>spatco.mix</td>
<td></td>
<td>720KB</td>
<td>9/17/2019 2:46 PM</td>
<td></td>
</tr>
<tr>
<td>spatco.prg</td>
<td></td>
<td>489KB</td>
<td>9/17/2019 2:46 PM</td>
<td></td>
</tr>
<tr>
<td>spatco_app.mapp</td>
<td></td>
<td>195KB</td>
<td>9/17/2019 2:46 PM</td>
<td></td>
</tr>
<tr>
<td>spatco_app.pj</td>
<td></td>
<td>3 KB</td>
<td>9/18/2019 10:48 AM</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_39b.slx</td>
<td></td>
<td>22 KB</td>
<td>9/17/2019 2:46 PM</td>
<td></td>
</tr>
<tr>
<td>therm_39b.svcm</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

[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)
This is a zero-noise situation. I expect the data to look like this.
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.
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.
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.
Retrieving files from repository:

Receiving objects

[Stop button]
SpatCo Thermostats

\[ [t_{cmd}, t_{measured}, t_{actual}] = \text{get Therm Data}(sys); \]

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

\[ \text{plot Therm Data}(t_{cmd}, t_{actual}); \]

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
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([num2str(nnz(maxIndices))])
hold off
legend
```
function [t_cmd, t_measured, t_actual] = get_therm_data(sys)

% Set up the simulation

tFinal = 100;

sinin = simin.SimulationInput('therm_19b');
sinin = simin.setModelParameters('StartTime', '0', 'StopTime', num2str(tFinal));
% sinin = simin.setModelParameters('therm_19b:StateSpaceA', 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)) + 10;
t_cmd = repmat(t, length(t), 1, cmd);

do = 0.1;
newTime = hours(0.1:tFinal);

% Simulate

% Run the simulation

sinin = simin.setExternalInput([hours(t_cmd, newTime), t_cmd.cmd]);

sinout = sim(sinin);

% Find local maxima

maxIndices = findlocalmax(PeakSig);

% Visualize results

clf
plot(PeakSig, 'Color',[160 85 226]/255, 'DisplayName', 'Input data');
hold on
plot(find(maxIndices), PeakSig(maxIndices), 'r', 'Color',[177 83 25]/255, 'DisplayName', 'Local maxima');
title(['Number of extrema: ' num2str(max(maxIndices))]);
hold off
legend

ybad = t_actual.y;

% noise

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

% drop-outs

imbad = rand(size(ybad)) > 0.8;
ybad(imbad) = NaN;

% missing

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

% measured = ybad;
Live Tasks

% Set up the simulation

```
tFinal = 100;

simin = Simulink.SimulationInput('therm_19b');
simin = simin.setModelParameter={['StartTime','0','StopTime',num2str(tFinal)});

% simin = simin.setModelParameter('therm_19b\StateSpace\A',sys.A)

% Pick some random times to change the commanded temperature

t = sort(tFinal*rand(10,1));
t(1) = 0;

cmd = randi([0, size(t)]); + 10;
t_cmd = timetable(hours(t),cmd);

dt = 0.1;

newTime = hours(1:ceil(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

Select input data: hertz

Define local extrema

Chart type: max

Mx search type: interior

Visualize results

Local extrema: yes

```
ybad = t_actual.y;

% noise

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

% drop-outs

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

% missing

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

t_measured.y = ybad;
```
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.
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.
```
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
   plot_therm_data(t_cmd, t_actual)

2. 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

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

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

Clean Outlier Data

Find, fit, or remove outliers

Select data

Input data: select

Specify cleaning method

Cleaning method: Fill outliers

Define outliers

Detection method: Median

Threshold factor: 3

Visualize results

Filled outliers, Cleaned data, Input data, Outliers, Outlier thresholds, 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

Find, fill, or remove outliers

Select data

- Input data
- X-axis

Specify

- t_actual
- Cleaning up t_cmd
- t_measured
- Define outliers

Detection method

- Median
- Standard factor

Visualize results

- Filled outliers
- Cleared 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.
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

```matlab
(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: Median, Threshold factor: 3

Visualize results

- Filled outliers, Cleared data, Input data, Outliers, Outlier thresholds, Outlier center

Thermostat Data

Number of outliers: 79
linear interpolation method

Number of outliers: 79

- Input data
- Cleaned data
- Outliers
- Filled outliers
- Outlier thresholds
Number of outliers: 79

Input data
- Cleaned data
- Outliers
- Filled outliers
- Outlier thresholds
Linear interpolation method

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, y
- **X-axis** : Time
- **Cleaning method** : Tuber and Linear interpolation
- **Define outliers** : Median
- **Threshold factor** : 3

Visualize results:
- Filled outliers
- Cleared 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) = Filled outliers in t_measured_y using the linear interpolation method

1. **Select data**
   - Input data: t_measured, y, X-axis: t_measured, Time

2. **Specify cleaning method**
   - Cleaning method: Fill outliers, Linear interpolation

3. **Define outliers**
   - Detection method: Median, Threshold factor
   - Visuallization result: Mean, Moving median

---

**Thermostat Data**

- Number of outliers: 79
- Input data: Measured, Cleaned
- Outliers: 79
- Filled as above: 79
- Outlier threshold: Moving mean

---

**Number of outliers:**

- 79
- Filled as above: 79
- Outlier threshold: Moving mean
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) = Cleaned outliers in t_measured using the linear interpolation.

Select data:
- Input data: `t_measured`
- Y-axis: `t_actual`

Specify cleaning method:
- Cleaning method: `Linear interpolation`

Define outliers:
- Detection method: `Threshold factor`
- Threshold factor: `2.5`
- Moving window: `Centred`
- Units: `Days`

Visualize results:
- Filled outliers
- Cleaned data
- Input data
- Outliers
- Outlier detection
- Thresholds

Number of outliers: `79`
SpatCo Thermostats

1. \([t\_cmd, t\_measured, t\_actual] = \text{get\_therm\_data}(\text{sys});\)

   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.

---

**Clean Outlier Data**

(cleanedData) = Filled outliers in t\_measured\_y using the linear interpolation method

- **Select data**
  - Input data \( t\_measured \)
  - \( y \) \( \rightarrow \) \( x \)-axis

- **Specify cleaning method**
  - Cleaning method: Fill outliers \( \rightarrow \) Linear interpolation

- **Define outliers**
  - Detection method: Moving median \( \rightarrow \) Threshold factor: 2.5

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

---

**Thermostat Data**

- **Temperature (y-axis):**
  - Measured
  - Cleaned

- **Time (x-axis):**
  - Range: 0 to 100

**Number of outliers:** 79
Clean Outlier Data

% Cleaned data 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, Sample points: t_measured.Time

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

% Fill outliers
[cleanedData, outlierIndices, thresholdLow, thresholdHigh] = ... filloutliers(t_measured, y, 'linear', 'movmedian', days(1), ... 'thresholdFactor', 2.5, 'samplePoints', t_measured.Time);

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

% Plot outliers
plot(t_measured.Time(outlierIndices), t_measured.y(outlierIndices), 'Color', [64 0 64]/255, 'DisplayName', 'Outliers')
title(['Number of outliers: ' num2str(size(outlierIndices))])

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

% Plot outlier thresholds
plot([thresholdHigh], missing: t_measured.Time), ...
Clean Outlier Data
(cleanedData) = Filled 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
Moving window Centered ▼ "Moving window" ▼ Units Days ▼

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

% fill outliers
(cleanedData, outlierIndices, thresholdLow, thresholdHigh) = ... fillOutliers(t_measured, y, 'linear', 'movmedian', 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.y, [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(length(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; missing], [145 146 147]/255,...
    'Color', [0 114 189]/255, 'LineWidth', 1.5, DisplayName, 'Thresholds')

Thermostat Data

Temperature/alog Ordinals

Number of outliers: 79
Clean Outlier Data

$cleanedData = \text{Filled outliers in } t\text{\_measured, } y \text{ using the linear interpolation method.}$

Select data

Input data $t\text{\_measured, } y, x \text{ \_measured, } \text{\_measured, } \text{\_Time.}$

Specify cleaning method

Cleaning method $\text{Fill outliers, Linear interpolation.}$

Define outliers

Detection method $\text{Moving median, Threshold factor 2.5.}$

Moving window $\text{Centered, 1 day.}$

Visualize results

$\text{Filled outliers, Cleaned data, Input data, Outliers, Outlier thresholds, Outlier center.}$

Visual results

clf
plot(t\_measured\_Time, t\_measured\_y, [100 180 255, 'Displayname', 'Input data'])
hold on
plot(t\_measured\_Time, cleanedData, [0 114 180 255, 'LineWidth', 1.5, 'DisplayName', 'Cleaned data'])

% Plot outliers
plot(t\_measured\_Time(outlierIndices), t\_measured\_y, [0 104 44 255, 'DisplayName', 'Outliers'])
title(['Number of outliers: ' num2str(outlierIndices)])

% Plot filled outliers
plot(t\_measured\_Time(outlierIndices), cleanedData(outlierIndices), [217 83 25 255, 'DisplayName', 'Filled outliers'])

% Plot outlier thresholds
plot([thresholdHigh; missing; thresholdLow; missing], t\_measured\_Time, [124 146 146 124, 'DisplayName', 'Thresholds'])
Clean Outlier Data

```matlab
function [cleanedData] = filloutliers(t_measured, y, linearInterpolationMethod)

% Select data
inputData = t_measured;
Yaxis = y;
Xaxis = t_measured;
Time = t_measured;

% Specify cleaning method
cleaningMethod = 'Fill outliers';
linearInterpolationMethod = 'Linear interpolation';

% Define outliers
thresholdFactor = 2.5;
SamplePoints = '
T_measured Time';

% Visualize results
figure;
plot(t_measured, y, 'color', [100 185 220]/255);
histc(t_measured, linspace(50, 90, 100), 'DisplayName', 'Input data');
hold on;
plot(t_measured, cleanedData, 'color', [0 114 189]/255, 'LineWidth', 1.5);

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

% Plot outlier thresholds
plot([thresholdHigh; missing; thresholdLow], t_measured, 'Color', [145 145 145]/255);
```

Thermostat Data

Number of outliers: 79
SpatCo Thermostats

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

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

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

   `cleanedData2 = FillOutliersInTMeasuredYUsingTheLinearInterpolationMethod;`
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.

```matlab
\texttt{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.
```

---

`mis`

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

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

   - Clean Missing Data
     - Find, fill, or remove missing data
     - Select data
       - Input data select ▼ X-axis default ▼
     - Specify method
       - Cleaning method fill missing ▼
     - Visualize results
       - ✓ Cleared data ✓ Filled missing entries

   - Number of outliers: 79
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.

- **CleanedData** - Filled outliers in \(t_{measured}\) 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: 533
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 y using the linear interpolation method
- cleanedData2 = Filled missing data in cleanedData using the linear interpolation method

**Smooth Data**

- smoothData = 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 = 0.3

Visualize results

- Smoothed data
- Input 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.

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.
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)
```

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` 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
Argument Validation

```plaintext
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

```plaintext
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
Argument Validation

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
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
```
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><code>mustBePositive(value)</code></td>
<td>value &gt; 0</td>
<td><code>gt</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeNonpositive(value)</code></td>
<td>value &lt;= 0</td>
<td><code>ge</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeFinite(value)</code></td>
<td>value has no NaN and no Inf elements.</td>
<td><code>isfinite</code></td>
</tr>
<tr>
<td><code>mustBeNonNan(value)</code></td>
<td>value has no NaN elements.</td>
<td><code>isnan</code></td>
</tr>
<tr>
<td><code>mustBeNonnegative(value)</code></td>
<td>value &gt;= 0</td>
<td><code>ge</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeNegative(value)</code></td>
<td>value &lt; 0</td>
<td><code>lt</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeNonzero(value)</code></td>
<td>value ~= 0</td>
<td><code>eq</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeGreaterThan(value,c)</code></td>
<td>value &gt; c</td>
<td><code>gt</code>, <code>isscalar</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeLessThan(value,c)</code></td>
<td>value &lt; c</td>
<td><code>lt</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeGreaterThanOrEqualTo(value,c)</code></td>
<td>value &gt;= c</td>
<td><code>ge</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeLessThanOrEqualTo(value,c)</code></td>
<td>value &lt;= c</td>
<td><code>le</code>, <code>isreal</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeNonempty(value)</code></td>
<td>value is not empty.</td>
<td><code>isempty</code></td>
</tr>
<tr>
<td><code>mustBeNonsparse(value)</code></td>
<td>value has no sparse elements.</td>
<td><code>issparse</code></td>
</tr>
<tr>
<td><code>mustBeNumeric(value)</code></td>
<td>value is numeric.</td>
<td><code>isnumeric</code></td>
</tr>
<tr>
<td><code>mustBeNumericOrLogical(value)</code></td>
<td>value is numeric or logical.</td>
<td><code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeReal(value)</code></td>
<td>value has no imaginary part.</td>
<td><code>isreal</code></td>
</tr>
<tr>
<td><code>mustBeInteger(value)</code></td>
<td>value == <code>floor(value)</code></td>
<td><code>isreal</code>, <code>isfinite</code>, <code>floor</code>, <code>isnumeric</code>, <code>islogical</code></td>
</tr>
<tr>
<td><code>mustBeMember(value,S)</code></td>
<td>value is an exact match for a member of S.</td>
<td><code>ismember</code></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
showTempData.m
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", 1)
```

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

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

showTempData("spatco_hq.csv",-1) | ![Error]

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

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

```plaintext
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(floor_index,room_temp,colors);
```
Specify the floor and visualize

```
floor_index = 9;

h = plot_spatco_floorplan(gca);

color_offices(floor_index,room_temp,colors);
```
Specify the floor and visualize

defloor_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_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_spatial_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

```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,colors);
```

Floor 17
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, colors);
```
Specify the floor and visualize

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

```matlab
floor_index = 18;
h = plot_space(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 = 18;  
h = plot_space(color_office);```

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

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

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

```plaintext
floor_index = 18;
```

`h = plot_something(color_office);`
Live Controls
Specify the floor and visualize

```matlab
floor_index = 18 ;
h = plot_spacco_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
Specify the floor and visualize

Floor Number 31
In which Ned makes an App with AppDesigner
>> appdesigner
Create a new 2-panel app that automatically resizes and refloors its layout to accommodate different screen sizes.
SpatCo HQ
Thermostats

Office Plan for Floor 7

Floor: 7
Office: 701
In which Joe creates a Web App.
SpatCo HQ
Thermostats

Office Plan for Floor 1

Floor 1
Office 101
Web Apps
Web Apps
Web Apps
Web Apps
Web Apps
Web Apps

MATLAB App
Create an app installation file to share your app with MATLAB users.

Web App
Create a deployed web app using MATLAB Compiler.

Standalone Desktop App
Create a standalone desktop application using MATLAB Compiler.
Web Apps
Web Apps

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

- Close dialog after packaging
  Edit Project
  Cancel Packaging
Open output folder.
Web Apps
Web Apps
Web Apps

MATLAB Web Apps

Office Heat Explorer

Thermal Analysis of SpatCo World Headquarters

Version 2.1

Patients Display

Ugly Data App

Easily clean up your ugly data

Version 1.0

Better Viscosity Calculator.cf

Date modified: 9/11/2019 1:23 PM

Type: CTF File

Size: 1.107 KB

Office-HeatExplorer.cf

Date modified: 9/25/2019 9:17 AM

Type: CTF File

Size: 2.052 KB

Pancake Density Converter.cf

Date modified: 9/25/2019 9:17 AM

Type: CTF File

Size: 1.550 KB

The Spatulator.cf

Date modified: 9/11/2019 2:38 PM

Type: CTF File

Size: 1.167 KB
Web Apps

MATLAB Web Apps

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

Ugly Data App
Easily clean up your ugly data
version 1.0
SpatCo HQ Thermostats

Office Plan for Floor 1
Hi Ned:

The web app is complete.

Check it out at

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

Joe.
One week later...
SpoonCorp
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!