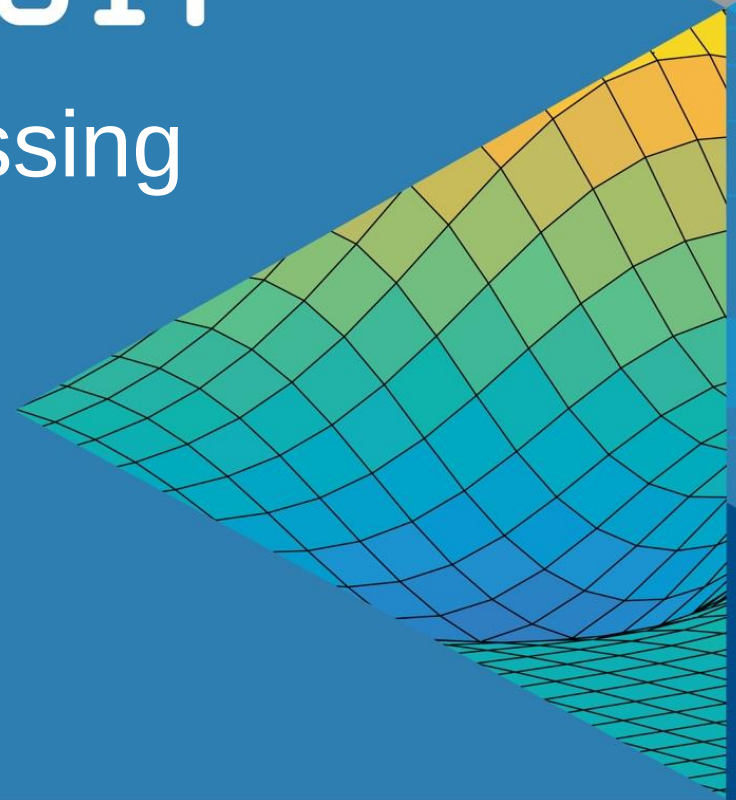
MATLAB EXPO 2017

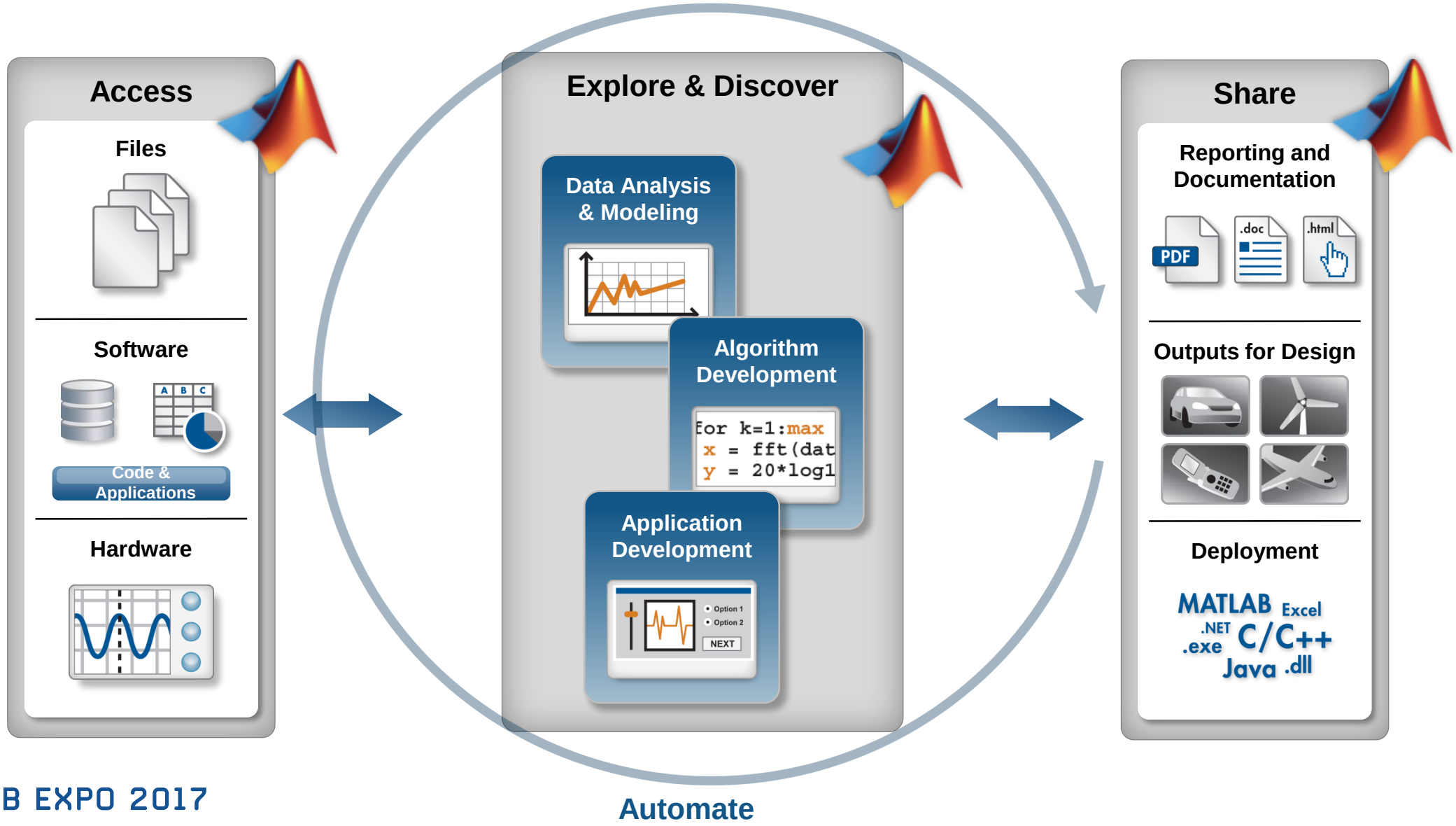
Introduction to Signal Processing

Modifying, measuring and extracting features from signals

Steven Thomsett



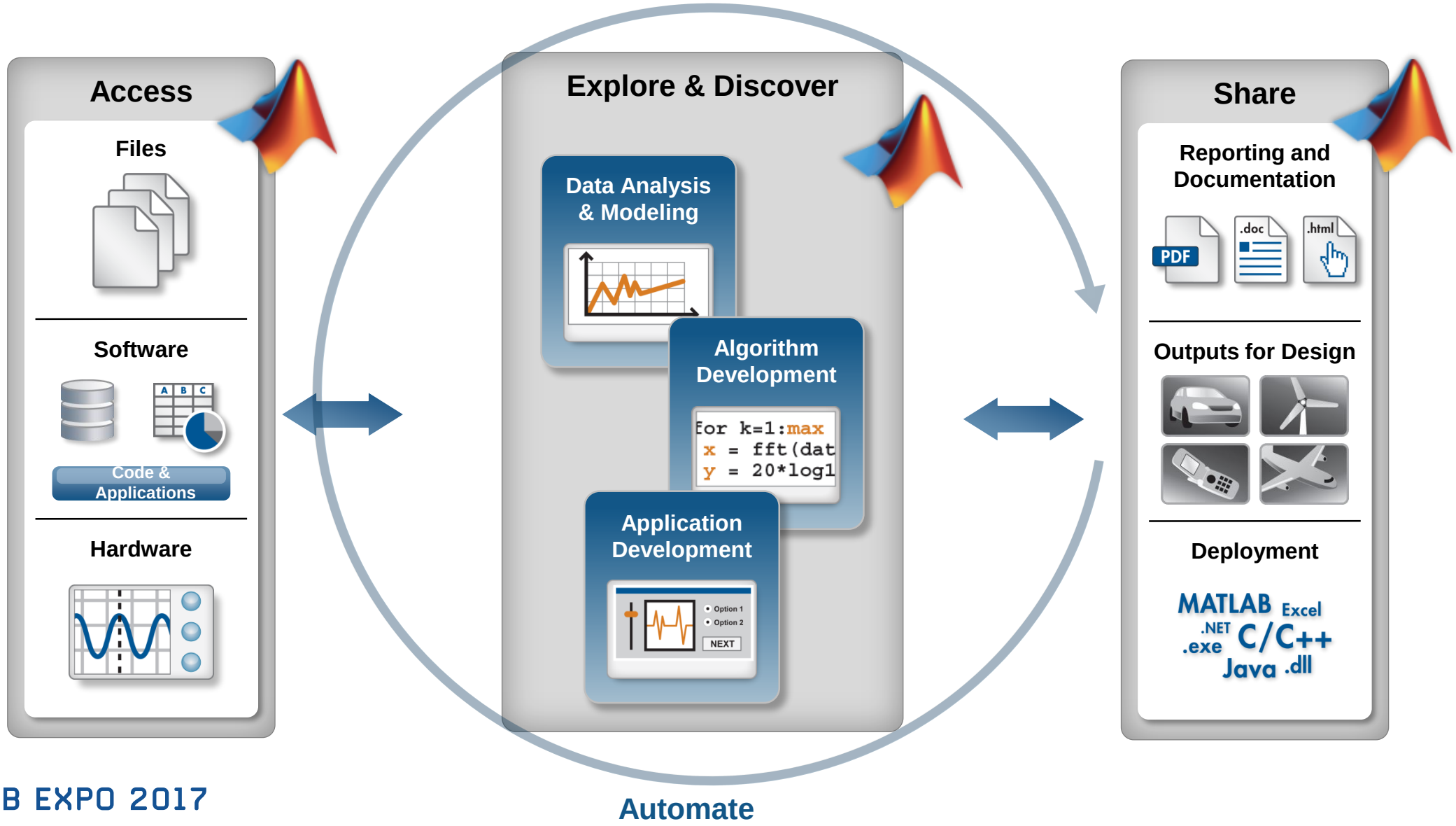
Technical Computing Workflow



Key Takeaways

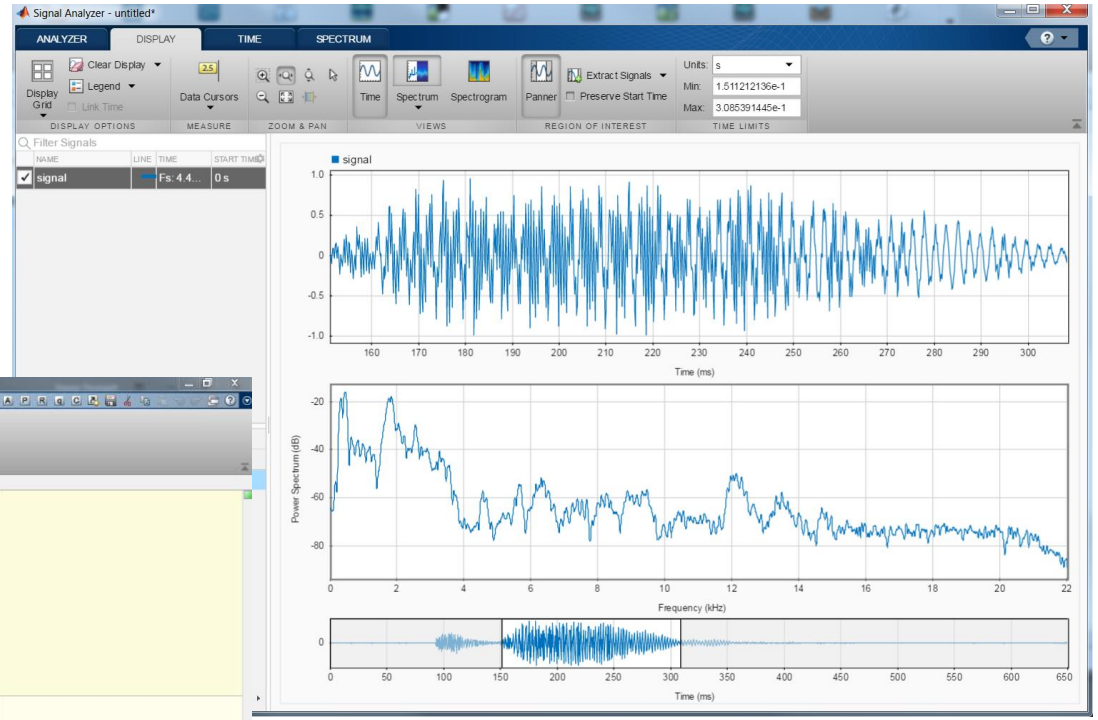
- Many signal processing techniques are common across workflows and applications
- MATLAB accelerates algorithm exploration with apps and common functions
- MATLAB provides the framework to transition from exploration to implementation

Technical Computing Workflow



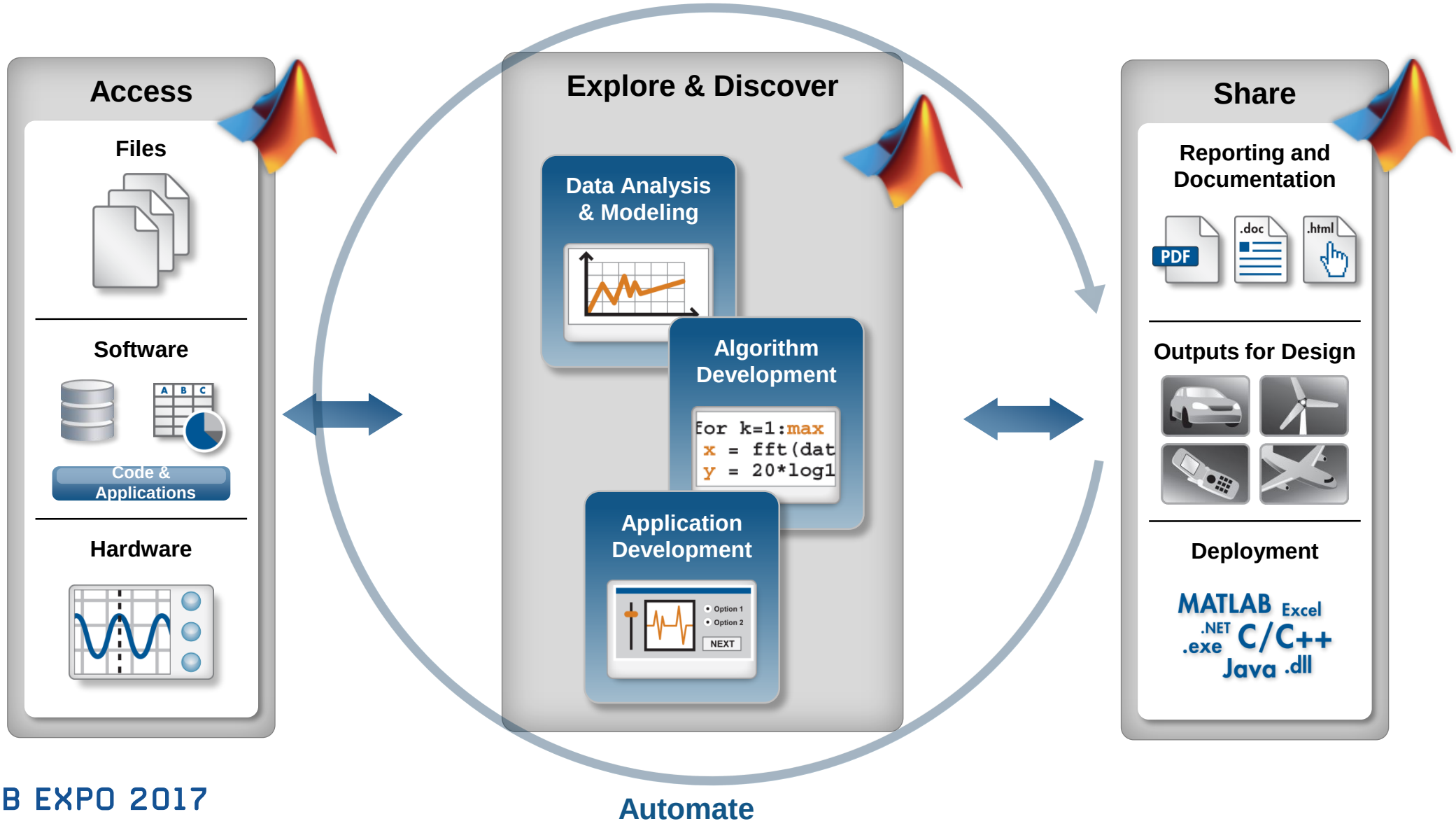
Exploring a signal interactively

- Use apps for interactive exploration
- Automatically generate code for automation
- You can modify and extend this for reuse or further analysis




```
% Compute power spectrum

% Generated by MATLAB(R) 9.3 and Signal Processing Toolbox 7.5.
% Generated on: 21-Aug-2017 13:57:57

% Parameters
timeLimits = [0.1794564 0.2219593]; % seconds
frequencyLimits = [0 22050]; % Hz

%%
% Index into signal time region of interest
signal_ROI = signal(:);
sampleRate = 44100; % Hz
startTime = 0; % seconds
minIdx = ceil(max((timeLimits(1)-startTime)*sampleRate,0))+1;
maxIdx = floor(min((timeLimits(2)-startTime)*sampleRate,length(signal_ROI)-1))+1;
signal_ROI = signal_ROI(minIdx:maxIdx);

% Compute spectral estimate
% Run the function call below without output arguments to plot the results
[Psignal_ROI, Fsignal_ROI] = pspectrum(signal_ROI,sampleRate, ...
    'FrequencyLimits',frequencyLimits);
```

Technical Computing Workflow



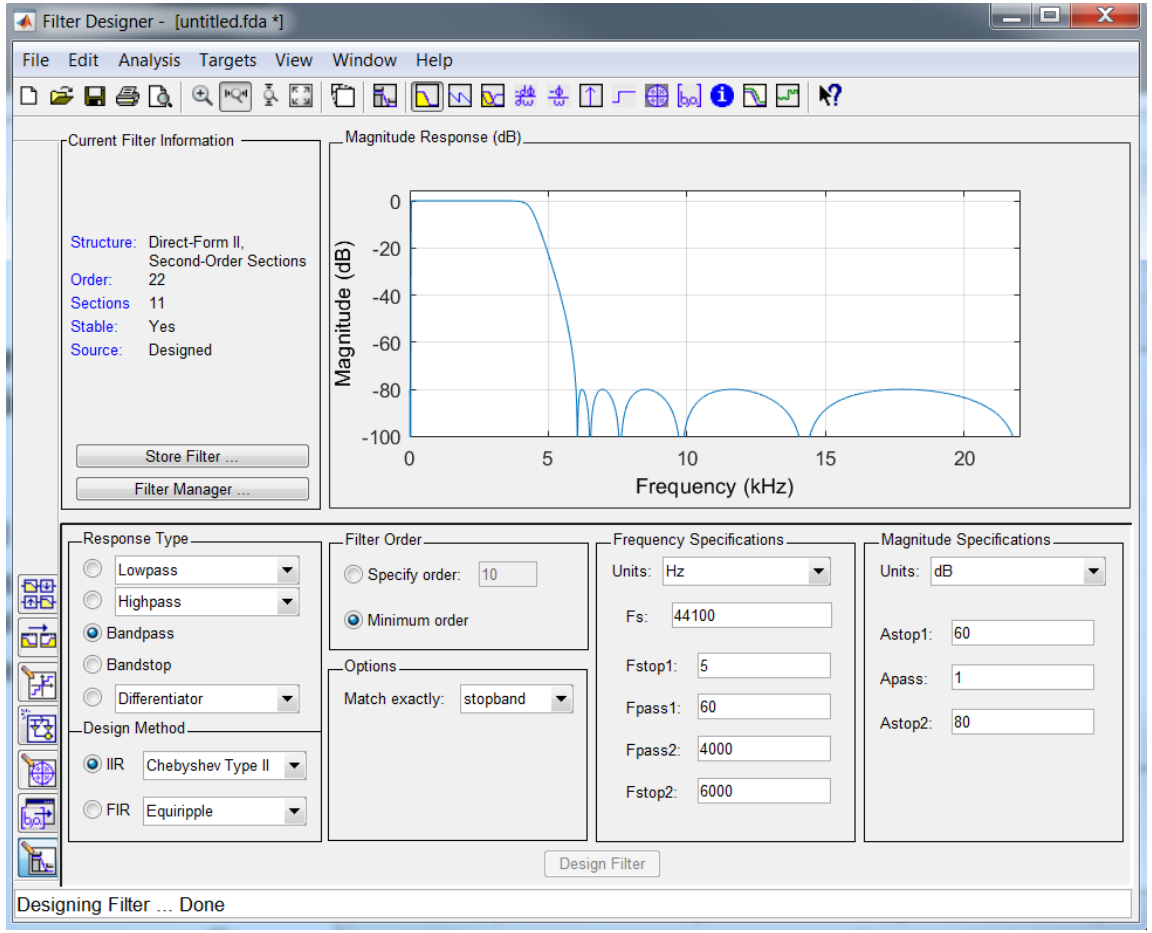
Design filters based on specifications

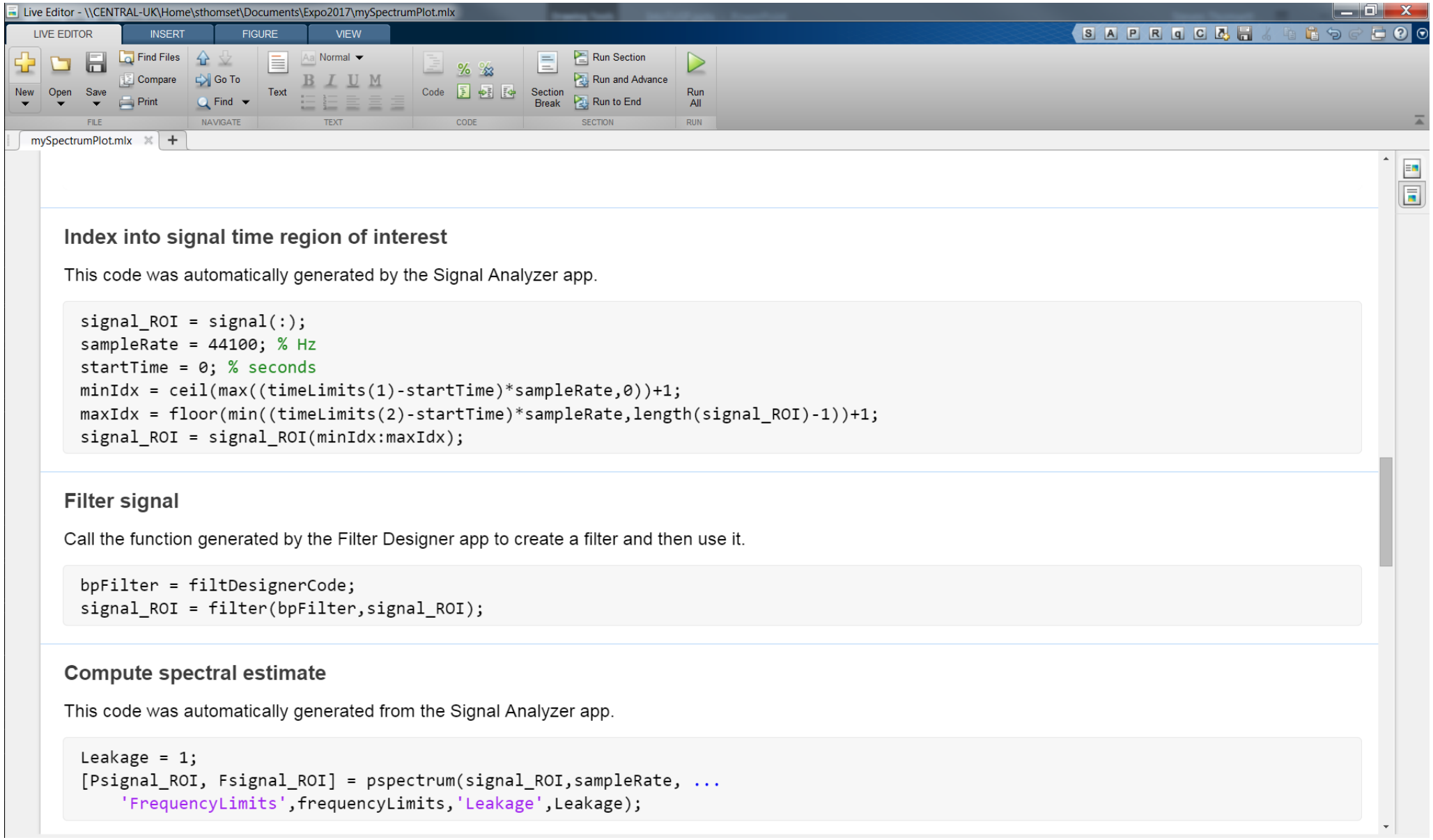
Interactively design filters based on specifications

- Can try settings and see the response immediately
- Generate MATLAB function when you are happy

Filter signal

```
bpFilter = filtDesignerCode;
signal_ROI = filter(bpFilter,signal_ROI);
```




Live Editor - \\CENTRAL-UK\Home\sthomset\Documents\Expo2017\mySpectrumPlot.mlx
LIVE EDITOR
INSERT
FIGURE
VIEW
New
Open
Save
Find Files
Compare
Print
Go To
Find
Text
Normal
Code
Section Break
Run Section
Run and Advance
Run to End
Run All
FILE
NAVIGATE
TEXT
CODE
SECTION
RUN
mySpectrumPlot.mlx
Index into signal time region of interest
This code was automatically generated by the Signal Analyzer app.
signal_ROI = signal(:);
sampleRate = 44100; % Hz
startTime = 0; % seconds
minIdx = ceil(max((timeLimits(1)-startTime)*sampleRate,0))+1;
maxIdx = floor(min((timeLimits(2)-startTime)*sampleRate,length(signal_ROI)-1))+1;
signal_ROI = signal_ROI(minIdx:maxIdx);
Filter signal
Call the function generated by the Filter Designer app to create a filter and then use it.
bpFilter = filtDesignerCode;
signal_ROI = filter(bpFilter,signal_ROI);
Compute spectral estimate
This code was automatically generated from the Signal Analyzer app.
Leakage = 1;
[Psignal_ROI, Fsignal_ROI] = pspectrum(signal_ROI,sampleRate, ...
'FrequencyLimits',frequencyLimits,'Leakage',Leakage);

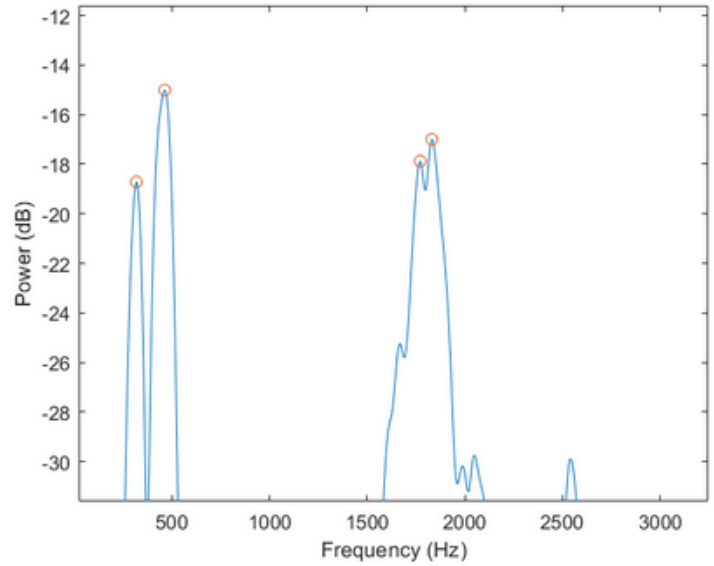
Extracting features and metrics from signals

Use common measurement techniques to accelerate development

- Detect features such as peaks and change points
- Extract metrics based on statistics or spectrum

Find peaks

```
[peakVal, peakLoc] = findpeaks(Psignal_ROI,'NPeaks',4,'SortStr','descend');
hold on
plot(Fsignal_ROI(peakLoc),db(peakVal,'power'),'o')

xlim([23 3246])
ylim([-31.6 -11.6])
```



Live Editor - \\CENTRAL-UK\Home\sthomset\Documents\Expo2017\mySpectrumPlot.mlx

LIVE EDITOR | INSERT | FIGURE | VIEW

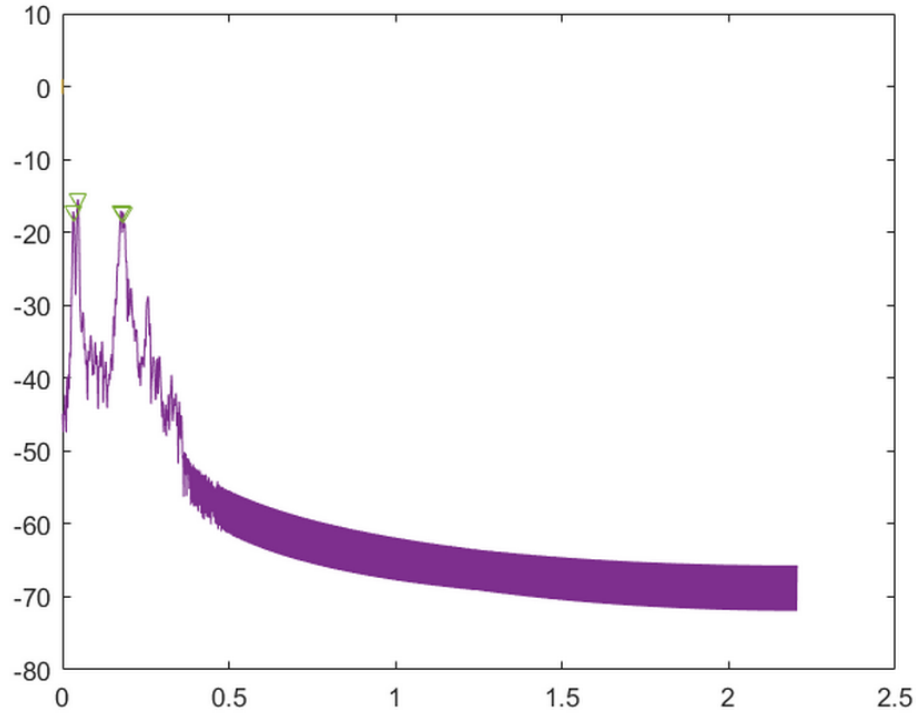
New | Open | Save | Find Files | Compare | Print | Go To | Find | Text | Normal | Code | Section Break | Run Section | Run and Advance | Run to End | Run All

mySpectrumPlot.mlx

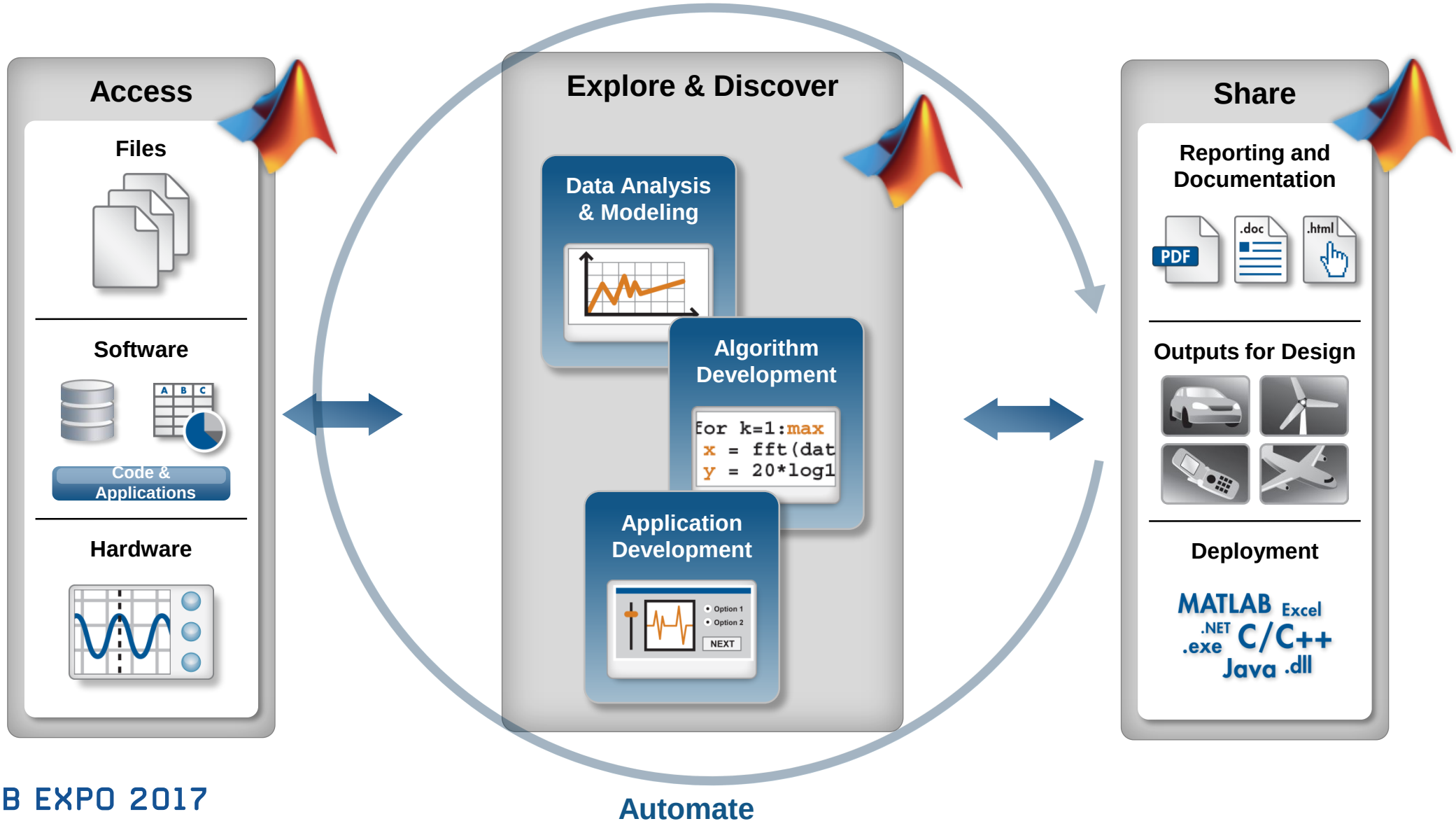
Find peaks

The command findpeaks can identify peaks and either plot or return the values for further analysis.

```
[peakVal,peakLoc] = findpeaks(db(Psignal_ROI,'power'),'NPeaks',4,'SortStr','descend');
peakFreq = Fsignal_ROI(peakLoc);
plot(Fsignal_ROI,db(Psignal_ROI,'power'),peakFreq,peakVal,'v')
```



Technical Computing Workflow



Considerations for transitioning to implementation

- Reusing components
- Configuring parameters
- Streaming data over time
- Automated triggers
- Integrating components

- Find out more about approaches to system modelling
  - Reusing and Prototyping Code to Accelerate Innovation: Smart Voice Interfaces 14:30
  - Introduction to Simulink and Stateflow 14:00

Gabriele Bunkheila, MathWorks

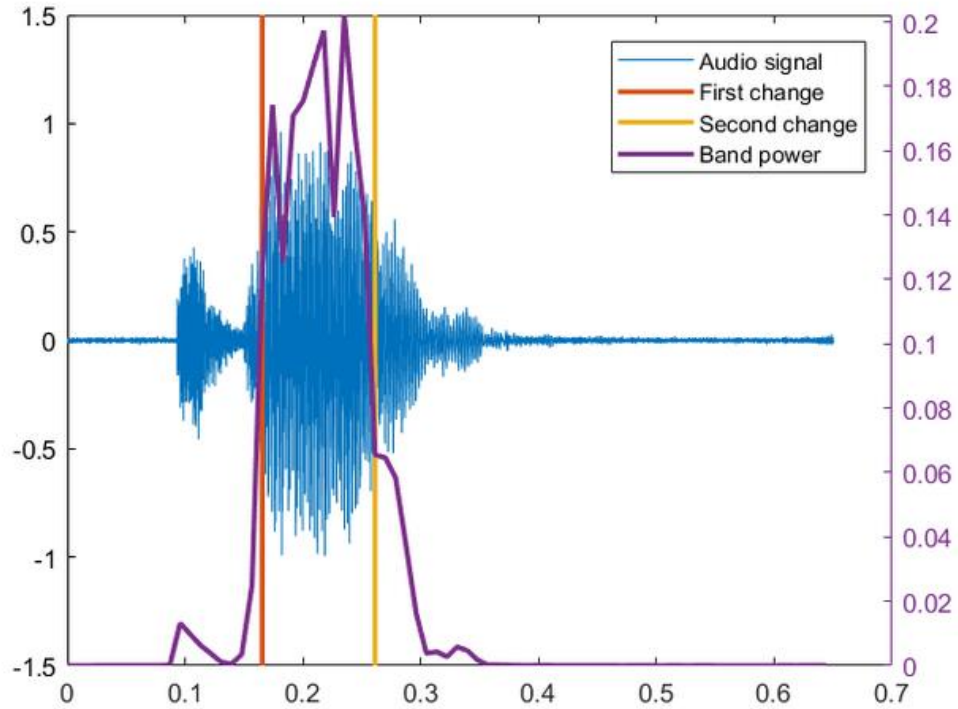
Jonathan Agg, MathWorks

Identify voiced speech

We can identify the transitions between voiced and unvoiced speech using changes in the power within the frequency band of interest.

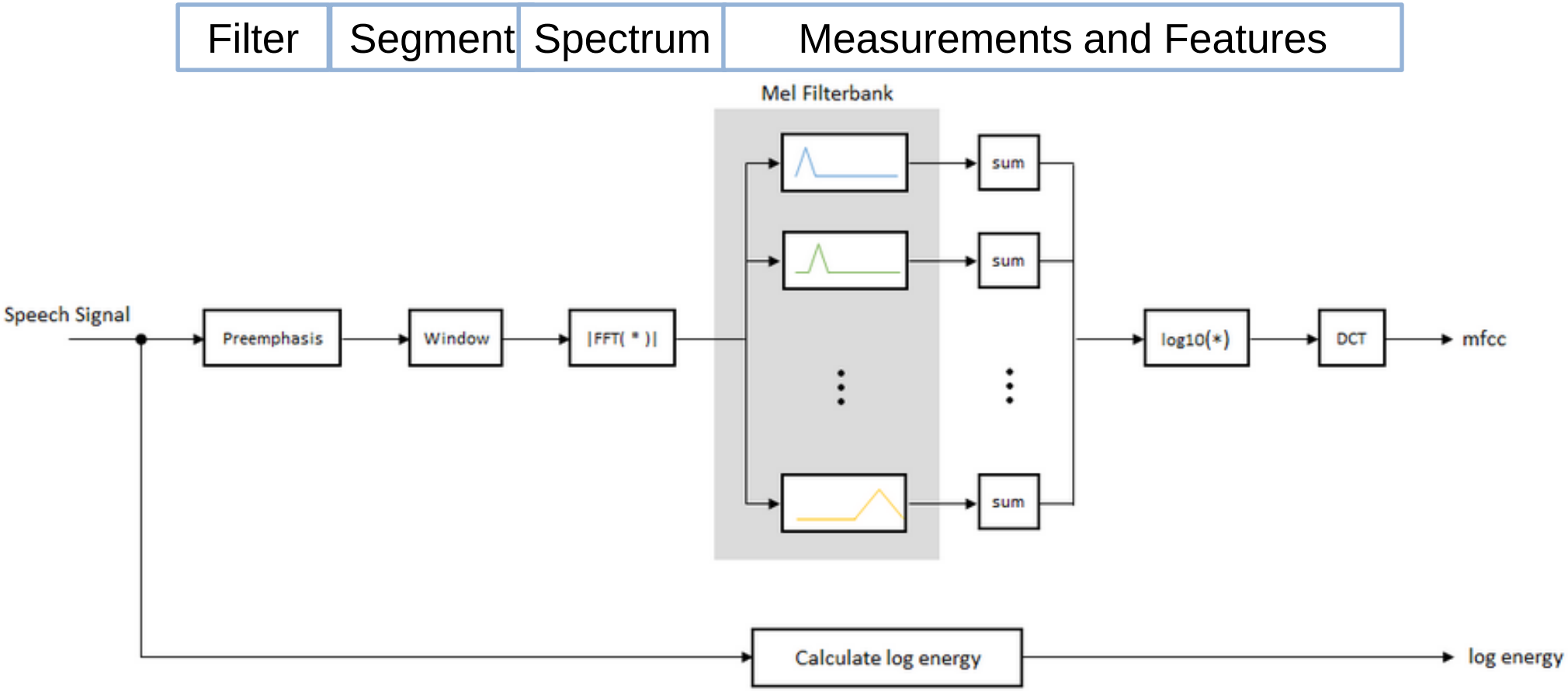
```
signalBuffer = buffer(signal,512,128);
signalBand = bandpower(signalBuffer,fs,[0 5000]);
changePts = findchangepts(signalBand,'MaxNumChanges',3);
changeTimes = (changePts-1)*384/fs;
timeLimits = changeTimes([1 2]); % seconds
```

signal	*20480x1 double*
signalBuffer	*512x75 double*



Speaker recognition algorithm

- Uses common techniques of filtering and spectral analysis to prepare data for measurements and feature extraction



Algorithm development workflow

- Interactive exploration of spectrum
- Generate code for automation
- Design filters based on specifications
- Extract features and measurements
- Moving from exploration to system models