Technical Computing Workflow

Access
- Files
- Software
- Hardware

Explore & Discover
- Data Analysis & Modeling
- Algorithm Development

Share
- Reporting and Documentation
- Outputs for Design
- Deployment

Automate

MATLAB EXPO 2017
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
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
Technical Computing Workflow

Access
- Files
- Software
- Hardware

Explore & Discover
- Data Analysis & Modeling
- Algorithm Development
  ```matlab
  for k=1:max
  x = fft(dat)
  y = 20*log1
  end
  ```
- Application Development
  - Option 1
  - Option 2
  - NEXT

Automate

Share
- Reporting and Documentation
- Outputs for Design
- Deployment
  - MATLAB
  - Excel
  - C/C++
  - Java
  - .exe
  - .dll
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

```matlab
bpFilter = filtDesignerCode;
signal_ROI = filter(bpFilter,signal_ROI);
```
Index into signal time region of interest

This code was automatically generated by the Signal Analyzer app.

```matlab
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.

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

Compute spectral estimate

This code was automatically generated from the Signal Analyzer app.

```matlab
Leakage = 1;
[Psignal_ROI, Fsignal_ROI] = psppectrum(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

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

```matlab
[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')
```
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.

```matlab
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