(Modestly) Real-time Matlab: Prototyping your DSP solution

G. Wakefield

EECS 452

09OCT14

Scope

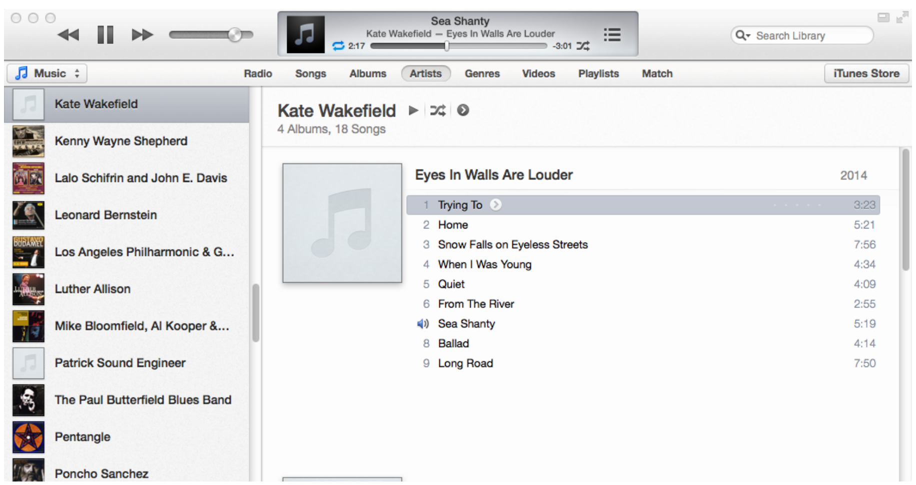
- Demos 1, 2, 3
- Build around an audio example
  - *Generalizes to any one-dimensional signal of interest*
- Techniques
  - *fdatool: how to integrate into your own algorithm development*
  - *DSP toolbox: real time support at the function and object level (no block diagram/Simulink)*
  - *GUI programming: how to integrate user response into your system development and testing*
- Resources
  - *(Judicious) browsing through Matlab's documentation*

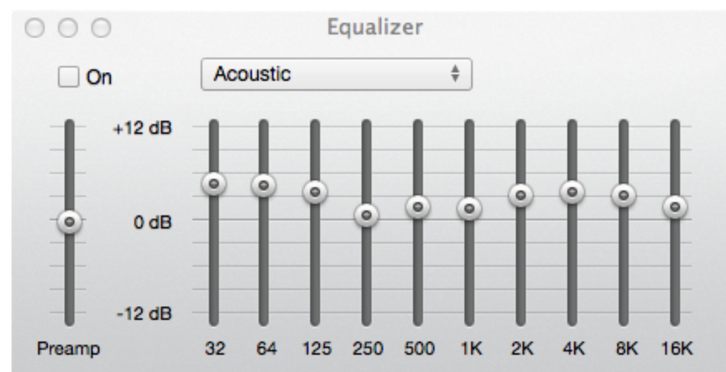
Toy Problem

- Implement a 3-band audio equalizer in Matlab.
  Your solution should provide a means for the user to load their own audio files, play them back, and adjust the gains of the equalizer bands in real time.

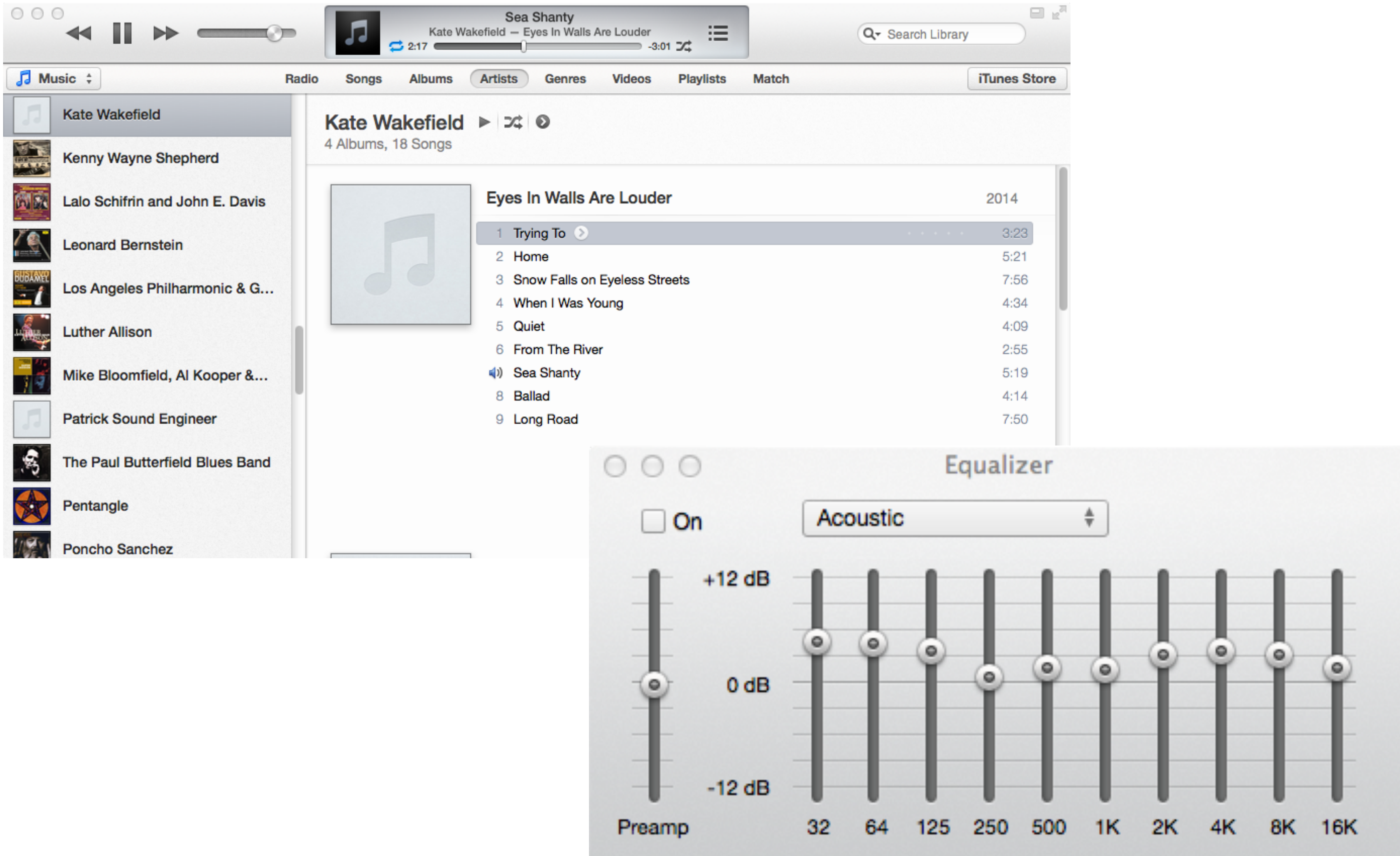
Reality Checking

- *Never solve a problem for which you haven't studied the prior art.*
  - Use your favorite music app as a model.

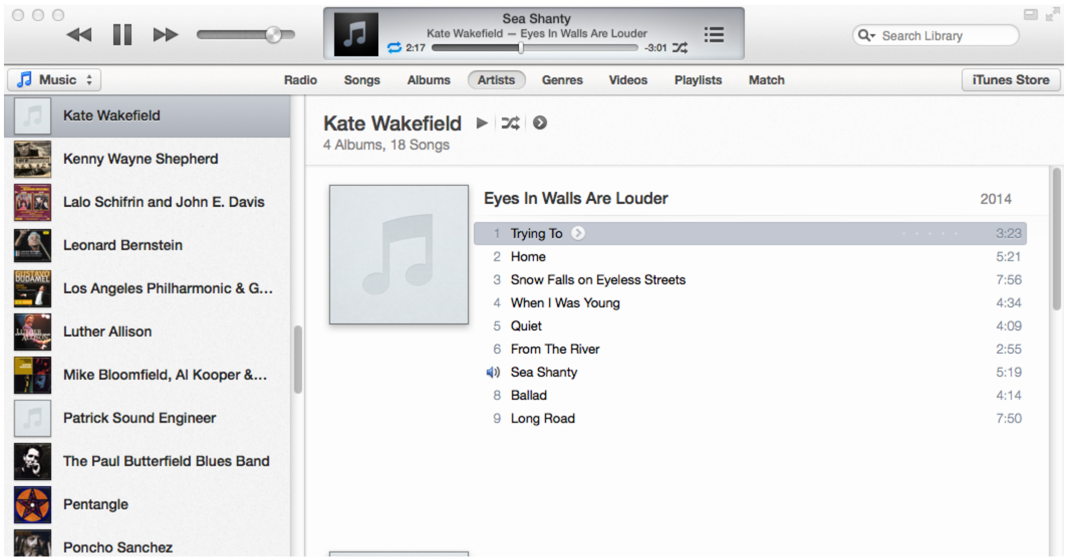




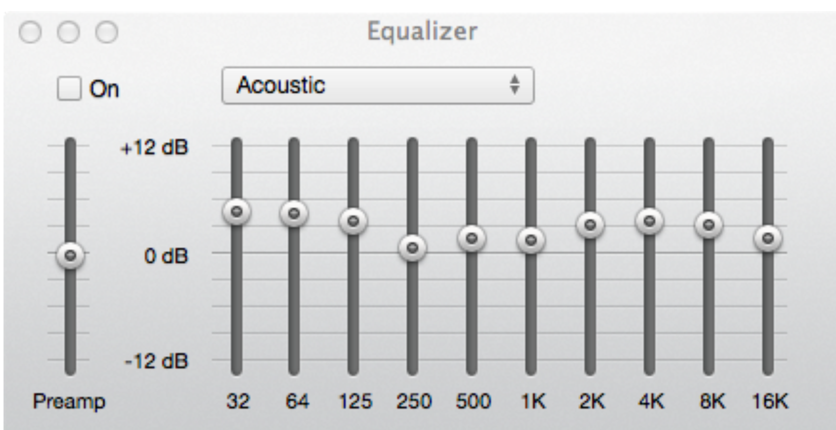
Particulars



Particulars



- Selectable file(s)
- Playback controls
- Output level control

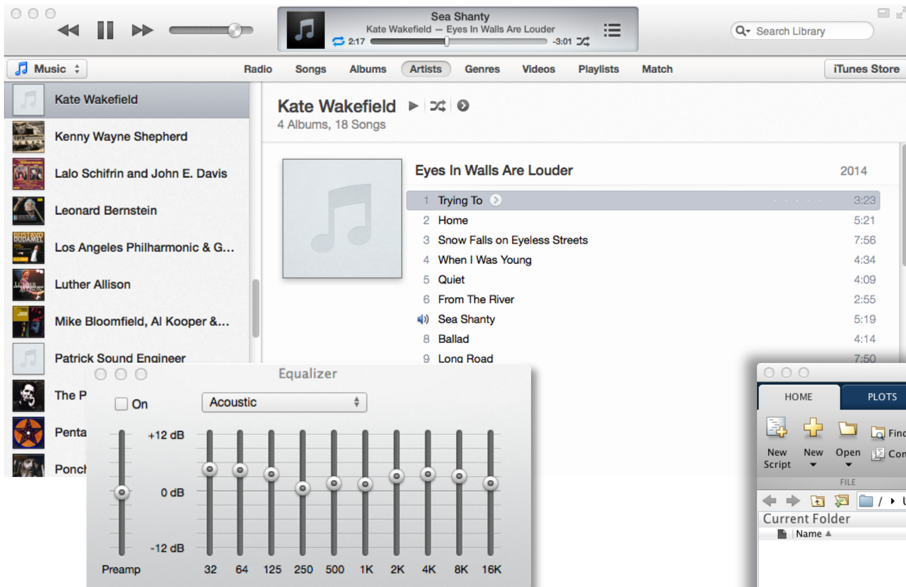


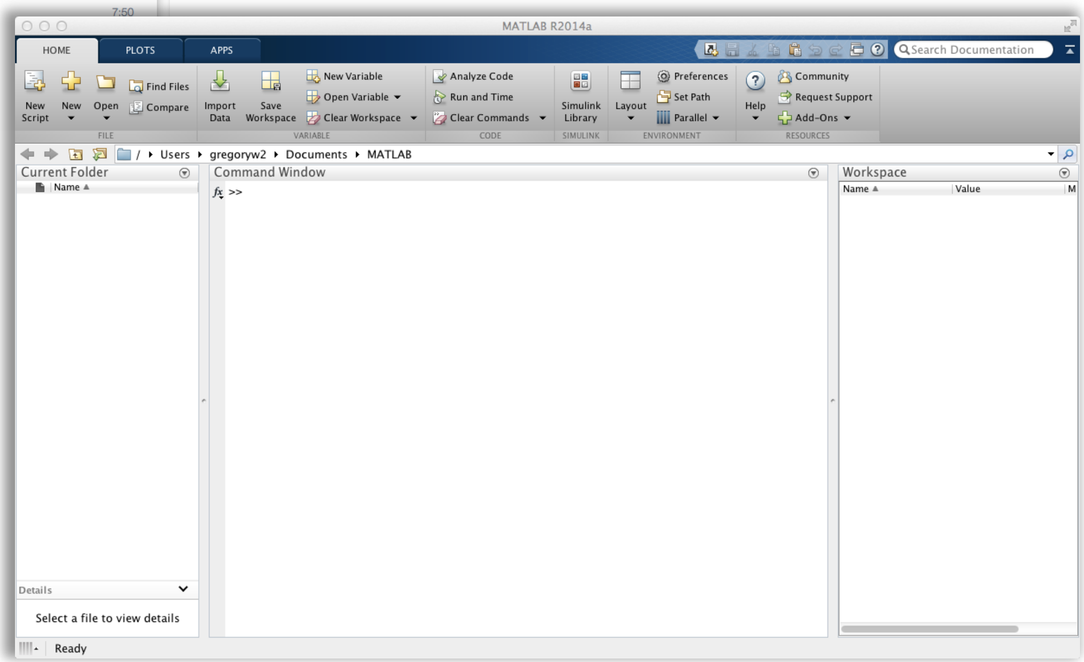
- Bandpass filters
- Parallel (filterbank) construction
- Range of gains (-12 to +12 dB)
- Center frequency organized *logarithmically*

Toy Problem (Details)

- Implement a 3-band audio equalizer in Matlab. *Your solution should provide a means for the user to load their own audio files, play them back, and adjust the gains of the equalizer bands in real time.*
  - Lowpass (Bass): 320 Hz cutoff
  - Bandpass (MidRange): 320 – 1280 Hz
  - Highpass (Treble): 1280 Hz cuton

Toy Problem (Details)





As far as the rest goes? Hmmmm…

Demo 1: Filtering and streaming

- *outsig* = filter(*filter spec*, *insig*)

- Matlab support
  - Audio i/o: standard and real-time
    - audioreader
    - audioplayer
    - audiowriter
  - Filter design
    - fdatool
    - exporting to the workspace and storing
  - Filter object
    - *state* playing the role of *memory* when crossing frame boundaries

Demo 1: Filtering and streaming

- demo1_a.m
  - *audioread, audioplay*
- fdatool
  - *Design lowpass, bandpass, highpass filters to meet specs, export as filter objects, save in MAT file*
- demo1_b.m
  - *Apply FIR designs*
- demo1_c.m
  - *Apply Butterworth IIR designs*
- demo1_d_dynamicEQ.m
  - *Naïve attempt to filter over blocks of the signal*
- demo1_(e…g)_debug_dynamicEQ.m
  - *Why we need a filter object to process dynamically over time*
- demo1_h_realtime_Audio.m
  - *DSP object level support for audio I/O: creating and destroying objects, the step and isDone methods*
- demo1_i_realtime_AudioEQ.m
  - *Putting all the pieces together in a partial solution to the original problem*

Demo 1: Notes

- *outsig* = filter(*filter spec*, *insig*)
  - *outsig has the same dimensions as insig (wrong in any real application)*
  - *break the filter operation into processing* frames *of insig*
    - *Efficient use of memory*
    - *Don't wait for batch processing to finish*
- Matlab support
  - Audio i/o: standard and real-time versions
    - audioreader: *supports a variety of commercial formats*
    - audioplayer: *supports a variety of playback rates and bit-depths*
    - DSP implementation: *output buffering dictated by size of sound card buffer and OS servicing times[*
    - Audiowriter: [not explored]
  - Filter design
    - fdatool: *"industrial strength" encapsulation of 40 years of research and development in digital filters*
    - exporting to the workspace and storing: *it's all about filter objects, not simply the numerator and denominator polynomial coefficients of the transfer function; use MAT files (or explore Mathwork's fda filter design management tool).*
  - Filter object
    - PersistentMemory: *state* playing the role of *memory* when crossing frame boundaries

Demo 2: GUI programming

- demo2_a.m
  - *load and play and audio file using pushbutton controls*
- demo2_b.m
  - *refine the user workflow of 2_a*
  - *taking advantage of OpeningFcn*
- demo2_c.m
  - *add "pause/resume" and "stop" functionality*
  - *interrupting the playback loop using "drawnow"*
- demo2_d.m
  - *add ability to adjust filter gains using "sliders"*
  - *avoiding extra event management – read state whenever possible (e.g., slider callbacks are factory stubs)*

Demo 2: GUI programming

- guide
  - *UIObjects: pushbutton, slider*
- anatomy of a Matlab GUI
  - Use *guide* to create visual layout
  - Program inside the factory-produced callback function stubs
  - Data structures to support GUI
- working within the "realtime loop"

Demo 2: GUI programming

- demo2_a.m/fig
  - *load and play and audio file using pushbutton controls*
- demo2_b.m/fig
  - *refine the user workflow of 2_a*
  - *taking advantage of OpeningFcn*
- demo2_c.m/fig
  - *add "pause/resume" and "stop" functionality*
  - *interrupting the playback loop using "drawnow"*
- demo2_d.m/fig
  - *add ability to adjust filter gains using "sliders"*
  - *avoiding extra event management – read state whenever possible (e.g., slider callbacks are factory stubs)*

Demo 2: Notes

- guide
  - *UIObjects: pushbutton, slider*
    manage visual display graphically; program the functionality within standard Matlab functions
- anatomy of a Matlab GUI
  - Use *guide* to create visual layout
  - Program inside the factory-produced callback function stubs
  - Data structures to support GUI
    handles = guidata(hObject) (get fresh copy)
    guidata(hObject,handles) (update handles)
- working within the "realtime loop"
  use uicontrols to update state information; minimize the queue of events outside the loop as much as possible

Demo 3: Systems Thinking

- Filter objects
  - Designing filters within your GUI rather than loading from an external file
- System objects
  - outsig = filter(Eq,insig)
  - Creating cascade and parallel systems
  - Modifying components of the system on the fly
- Working within the “realtime loop”
- Re-use old code: conceptual plug-ins

Demo 3: Systems Thinking

- demo3_a_preps, demo3_a.m/fig
  - *cascade and parallel objects: stages*
  - *updating attributes of stages on the fly*
  - *taking full advantage of OpeningFcn*
- demo3_b_preps, demo3_b.m/fig
  - *add filter design functionality to app*
- demo3_c.m/fig
  - *re-purpose sliders for variable (f,Q) comb filter*

Demo 3: Notes

- Filter objects
  - Designing filters within your GUI rather than loading from an external file
    Use fdesign to specify filter parameters; use design to create filter object. Rich collection of filter types – see fdesign documentation.
- System objects
  - outsig = filter(Eq,insig)
  - Creating cascade and parallel systems
  - Modifying components of the system on the fly
    Access components through stages; update filter coefficients while preserving state information; caveats abound: transient artifacts when filters change too quickly, requires filter order/structure remains invariant across updates.
- Working within the "real-time loop"
  An operational test of an algorithm that you intend to port to a stand-alone DSP. Provides a working environment in which to assess acceptable throughput delay (system responsiveness). Identify computational bottlenecks and evaluate design trade-offs.
- Re-use old code: conceptual plug-ins
  In principle, any time-varying linear system can be inserted into the basic structure of the demo3 apps. Compare demo3_a, b, c.

Additionals

- Latency
  - Audio playback: QueueDuration, BufferSize
  - GUI refresh (drawnow)
  - algorithm complexity
  - algorithm memory
  - demo4_latency exercises radiobutton alternative and manipulation of playback buffers
  - If it's such a problem…are there other solutions?
- How to use Matlab to prototype your DSP solution
  - Latency, complexity and frame rates
  - Sensitivity to finite-precision implementations, e.g., rather than loading DSP double-precision filters, load DSP single-precision filters of various forms
  - Evaluate design tradeoffs with respect to user criteria
  - Explore the catalog of solutions: DSP System , SP, System Identification, and IP Toolbox examples (to mention just a few). Other System Objects (scopes, spectrum analyzers, etc.)
- Caveats
  - Matlab implementation vs. DSP implementation
  - Clock the execution times for FIR and IIR in demo1_b and demo1_c. Compare with the complexity of each. Does this make sense?
  - Don't over- or under-design. Much is known about the sensory/motor constraints imposed by the user. Learn what these are.
  - Matlab should be running natively on your PC. Don't try to go through a CAEN network…