Eng. 100: Music Signal Processing
DSP Lecture 3 : 01/19
Project 1

Curiosity (animated sheet music):

- `http://www.youtube.com/watch?v=Rhv8iOYO8TY` (So What)
- `https://www.youtube.com/watch?v=lpc1lEJ-SRc`
  Jazzy Canon in D with transcription

Announcements:

- Lab 2 RQ's due 24 hours before first lab (last reminder)
- Homework 0 due Fri. 5 PM. Optional: "Introduction to Matrix Math"
- Read Project 1 before lab section next week. (reading questions...)
- `synth1-template.jl` on Google Drive (`dsp-projects`)
- Next three weeks: 3 DSP lectures related to Lab 3!

Using Piazza effectively

From http://16720.courses.cs.cmu.edu/piazza.html:

Questions are encouraged as a way to get help from other students when you are stuck, or feel like you're going down the wrong path. They are not meant as a way to solve a problem before you've struggled with it. Most learning comes from a little bit of struggling, and the assignments are meant to make you think. Before posting a question, ask yourself whether you're truly stuck or are just avoiding spending the time to figure it out. Struggling and debugging are a big part of learning in this (and any) class!

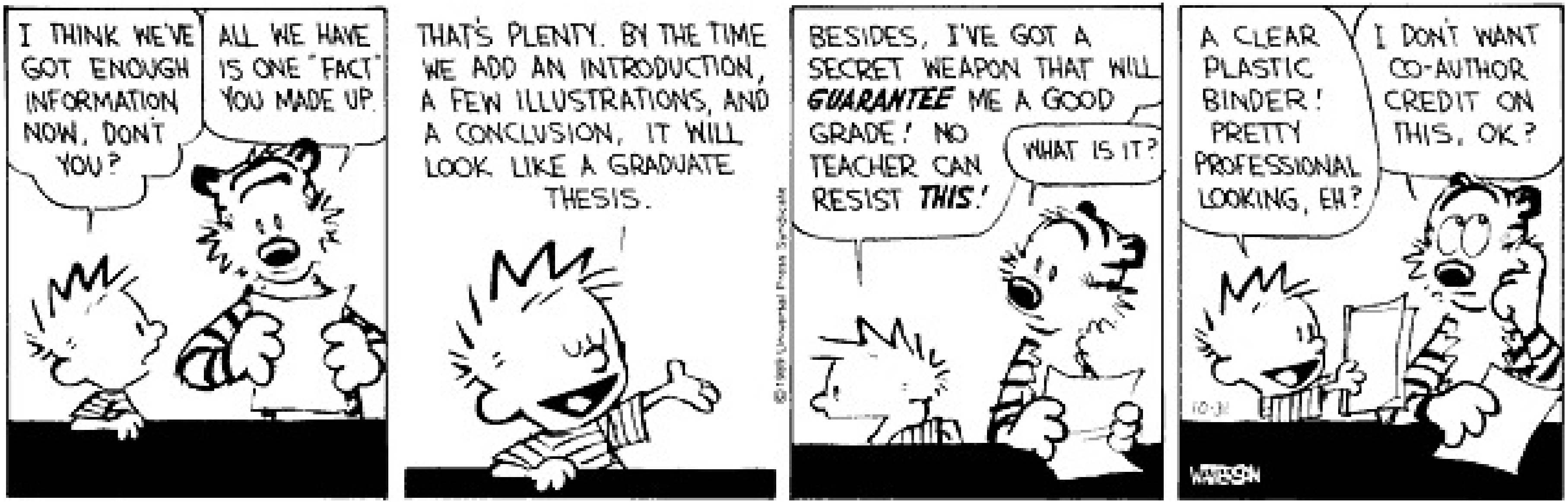
We encourage you to be an active participant and help your fellow students out by answering questions on Piazza. Avoid becoming overzealous and giving the answer though. Full lines of code should never be posted [publicly] on Piazza, and hints rather than solutions should be given. For example:
Question: "I tried subtracting the vector mu from the matrix A and MATLAB keeps breaking with the error message "...". I've tried several things and can't figure out how to make MATLAB subtract mu from each column."
Poor Answer: `bsxfun(@minus, A, mu)`
Good Answer: "Look into MATLAB's documentation/help on `bsxfun`. MATLAB's minus function only works on matrices that are the same dimension; you actually want to perform several subtraction operations on the matrix A."

Questions of the form "how do I solve this" might be best handled interactively 1-on-1 in office hours.

A more effective question should include some information about what you tried already. "I was trying to find the frequency using the arccos formula but I kept getting this error message [...] in Julia. Am I on the right track?"

We will try to answer clarification questions quickly.

Project 1: based on one fact...



Calvin: I think we've got enough information now, don't you?

Hobbes: All we have is one "fact" you made up.

Calvin: That's plenty. By the time we add an introduction, a few illustrations, and a conclusion, it will look like a graduate thesis. Besides, I've got a secret weapon that will guarantee a good grade! No teacher can resist this! A clear plastic binder! Pretty professional looking, eh?

Hobbes: I don't want co-author credit on this, OK?

Not one but two facts

- We can make basic music out of sinusoidal signals.
- We can determine the frequency of a pure sinusoidal signal using:

$$\boxed{f = \frac{S}{2\pi} \arccos\left(\frac{x[n+1]+x[n-1]}{2x[n]}\right).}$$

(Pure sinusoid sampled at $S \, \frac{\text{Sample}}{\text{Second}}$, using just 3 consecutive samples.)

Other key points from previous lecture:

- $\Omega = 2\pi f / S$ is the digital frequency of a sampled sinusoidal signal
- Models that arise frequently in music (and in perception, nature, engineering):

Exponential	$y = b\, a^x$	$\log(y) = \underbrace{\log(a)}_{\text{slope}} \; x + \underbrace{\log(b)}_{\text{intc.}}$	semi-log
Power	$y = b\, x^p$	$\log(y) = \underbrace{p}_{\text{slope}} \; \log(x) + \underbrace{\log(b)}_{\text{intc.}}$	log-log

Learning objectives

- Understand different music representations
- Understand MIDI formula
- Understand what synthesizers and transcribers do at high level
- Awareness of `Gtk` for GUI building
- Understand stem plotting using table lookup
- Be able to apply arccos method to sequence of notes
- Awareness of high level P1 specifications

Outline

Part 1: Music notes and notation

- Musical notes: frequencies and relations
- Basic musical staff and MIDI numbers

Part 2: Project 1

- Musical tone synthesizer and transcriber
- Specifications for the two deliverables
- `Julia` methods for graphical user interface (GUI)

Part 1: Music notes and notation

Music communication / representation

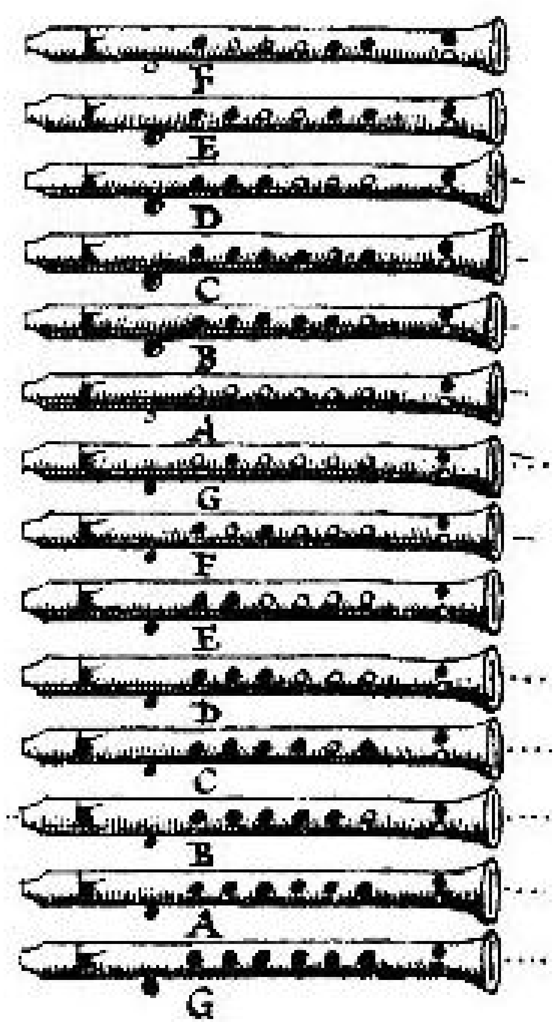
Lab 2 analyzes the frequencies of musical notes (numerical values like 261.6256Hz for middle C).

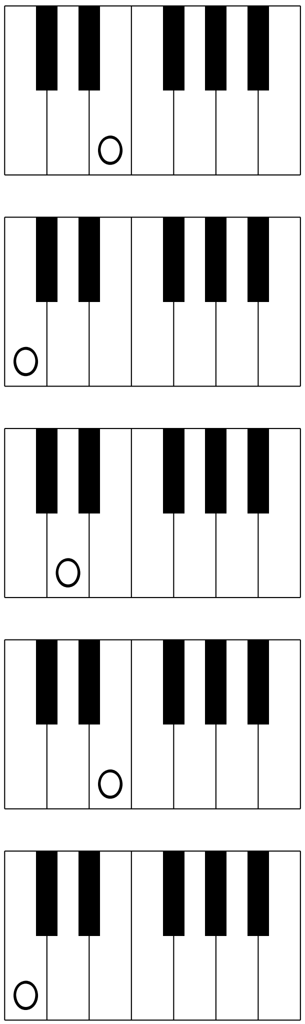
How does one person convey musical ideas to others?
A (technical?) communications problem!

Some options:

- Oral
- Pictures
- Video (?) (cf. animated sheet music example)
- Numbers
- Letters
- Words
- Diagrams / Symbols

Music communication: Pictures

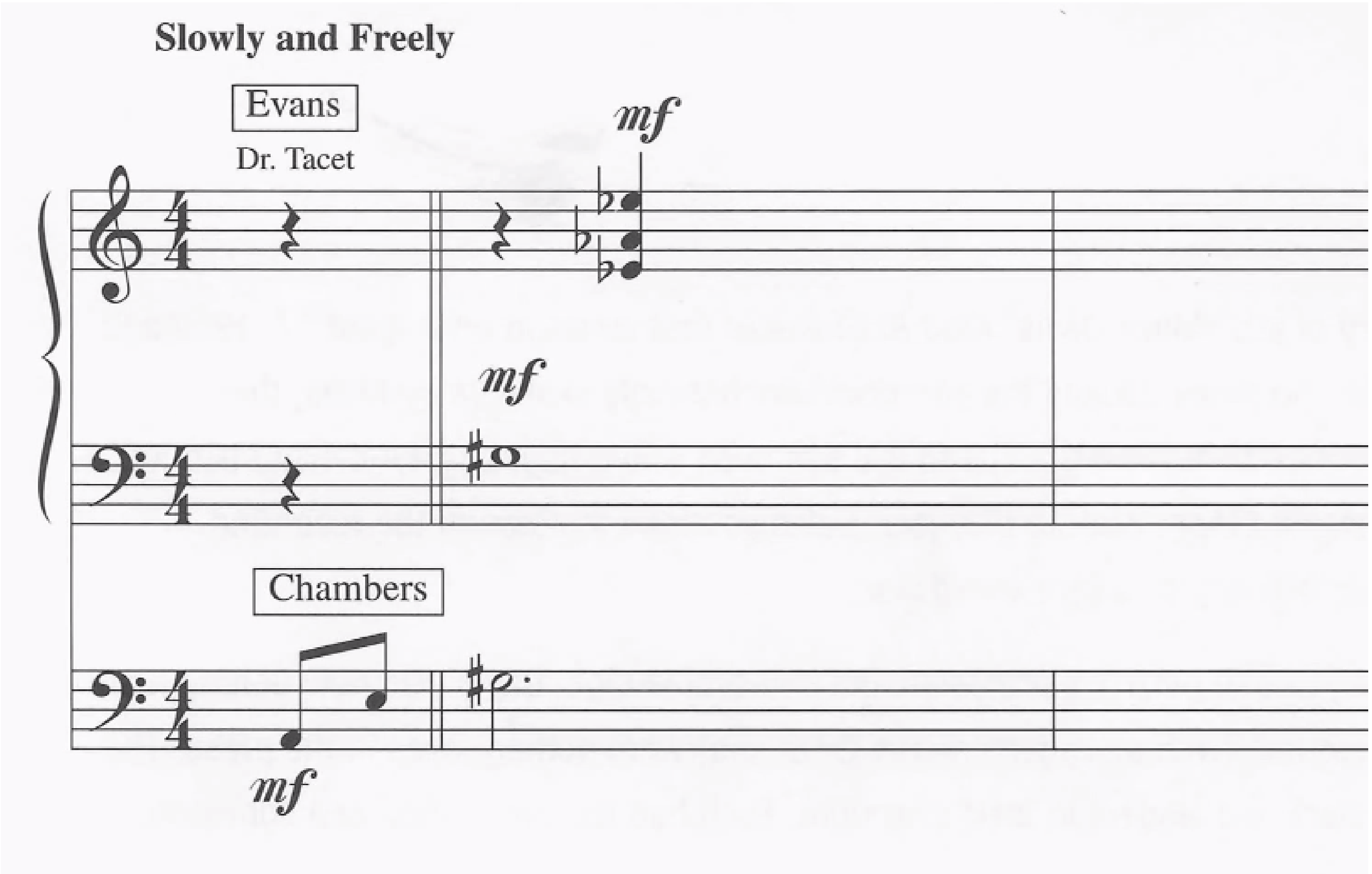




http://furulya.hu/historikus-ujjrendek/ujjrendek/Virgiliano.jpg

"The Victors" in pictures

Music communication: Video ?



Numerous YouTube "how to" videos to learn to play instruments.
Suboptimal for describing music to a musician.

Music communication: Numbers

List the frequencies, durations, and volume of each note?

"The Victors" as a list of frequencies, durations, volumes:

494	4	extra loud
392	2	loud
440	2	loud
494	2	extra loud
392	2	loud
440	2	loud
494	2	loud
523	4	loud
⋮	⋮	⋮

Very "physics" representation.
Reading this while performing would be hard.
Transposing (playing in another key) would be very hard.

Music communication: Numbers

- MIDI = Musical Instrument Digital Interface (defined 1982)
- Musical frequencies represented by integers (computer friendly)
- $\boxed{\text{MIDI number} = 69 + 12 \log_2(\text{frequency in Hertz}/440)}$
- "The Victors" represented in a computer via MIDI numbers:

$$(71, 67, 69, 71, 67, 69, 71, 72, \ldots)$$

- MIDI files also include amplitude (volume) and duration

Transposing (playing in another key) is easy:
just add or subtract an integer from all numbers.

MIDI: Designed by engineers!

- Defined "recently" (1982) by an engineer
- Designed by Dave Smith, "Father of MIDI"
- He chose this formula for the MIDI "pitch number:"

$$\text{MIDI} = 69 + 12 \log_2(\text{frequency in Hertz}/440)$$

- Why?
  - Allowed MIDI pitch numbers are 0,1,...,127
  - Lowest note on a piano A0 is 27.5 Hz, MIDI=21
  - A440, aka A4, just above middle C is 440 Hz, MIDI=69
  - Highest note on a piano C8 is 4186 Hz, MIDI=108
- What is the lowest possible MIDI frequency? $\boxed{??}$

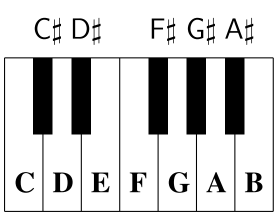
- Q0.1 Highest MIDI frequency (in Hz, nearest integer)?
  A: 440 B: 12544 C: 13290 D: 14080 E: 25088 ??

- Why 0 to 127? ??

Project 1 transcriber would be easier if the specification were to produce MIDI output...

Music communication: Letters

- 7 natural notes are named A,B,C,D,E,F,G.
- These include the 6 notes that appeared in "The Victors"
- These 7 occur the most frequently in many songs.
  cf. Morse code
  Aside: "CQD" abbreviation for "sécu" (from the French word sécurité) distress call,
  about 100 years before "OMG"
- These are the white keys on a piano (used be ivory)
- 5 "accidentals" labeled A♯,C♯,D♯,F♯,G♯ (black keys, used to be ebony)
- 5 equivalently labeled B♭,D♭,E♭,G♭,A♭
- A♯ ("A-sharp") same as B♭ ("B-flat"): between A and B



"The Victors" in letters: B G A B G A B C ...

Music communication: Words (skip)

Transposing between different keys requires different sequence of letters - not easy.

More than 1000 years ago (!), relative frequencies were named;
this communication system is called solfège or solfeggio.
cf. " The Sound of Music" score by Rogers and Hammerstein.

In the key of G, a "dictionary:"

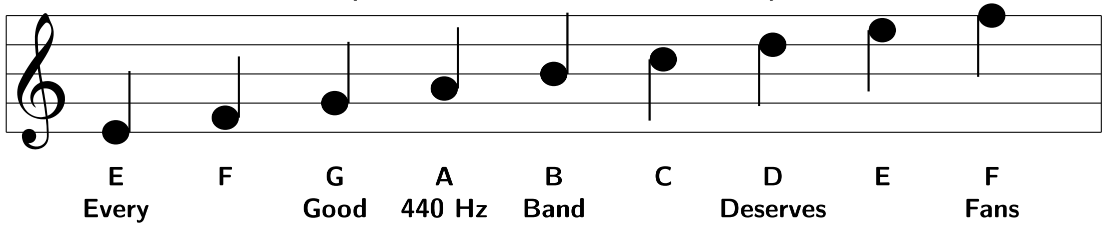
G	A	B	C	D	E	F$\sharp$	G
Do	Re	Mi	Fa	So	La	Ti	Do
392	440	494	523	587	659	740	784

In theory, by learning songs in solfège, you can play in any key.

"The Victors" in solfège: Mi Do Re Mi Do Re Mi Fa Do Re Mi ...

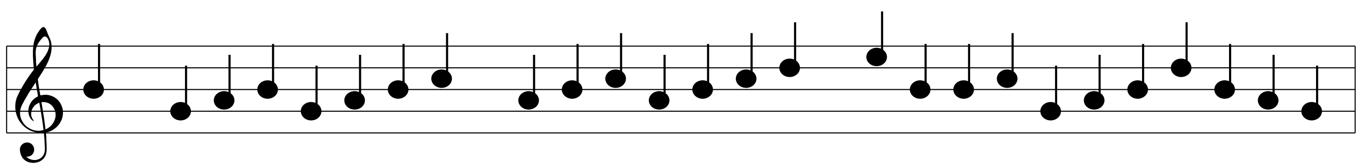
Music communication: Diagrams / Symbols

Musical staff representation (EGBDF on lines):



- The "natural" notes have the simplest symbols.
- Script "G" encircles line where "G" note is: treble clef or G-clef.
- (Accidentals are adorned with $\sharp$ or $\flat$.)
- A professional-grade transcriber must look much better, including ledger lines.

"The Victors" in (crude) musical staff notation:

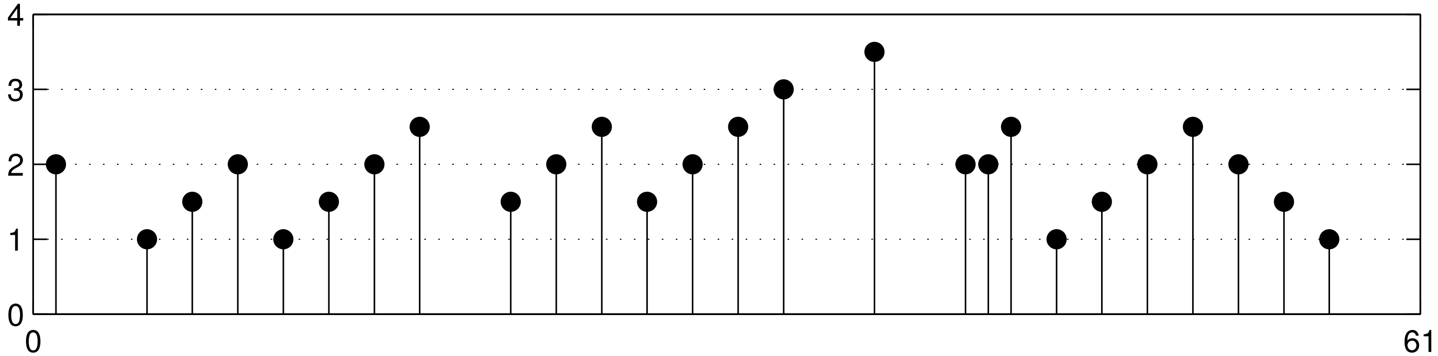


Higher frequencies are put higher. (But computer unfriendly format.)

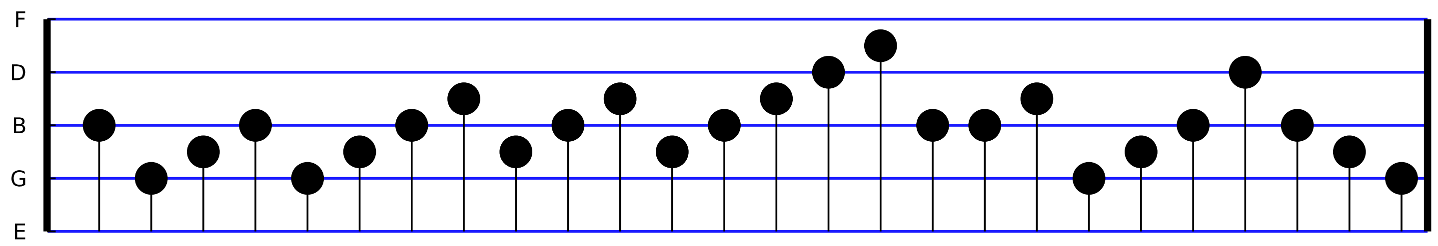
Music communication: `Julia` plots

`Julia` was designed for engineering, not for music notation!

"The Victors" in (crude) `Julia` plots:

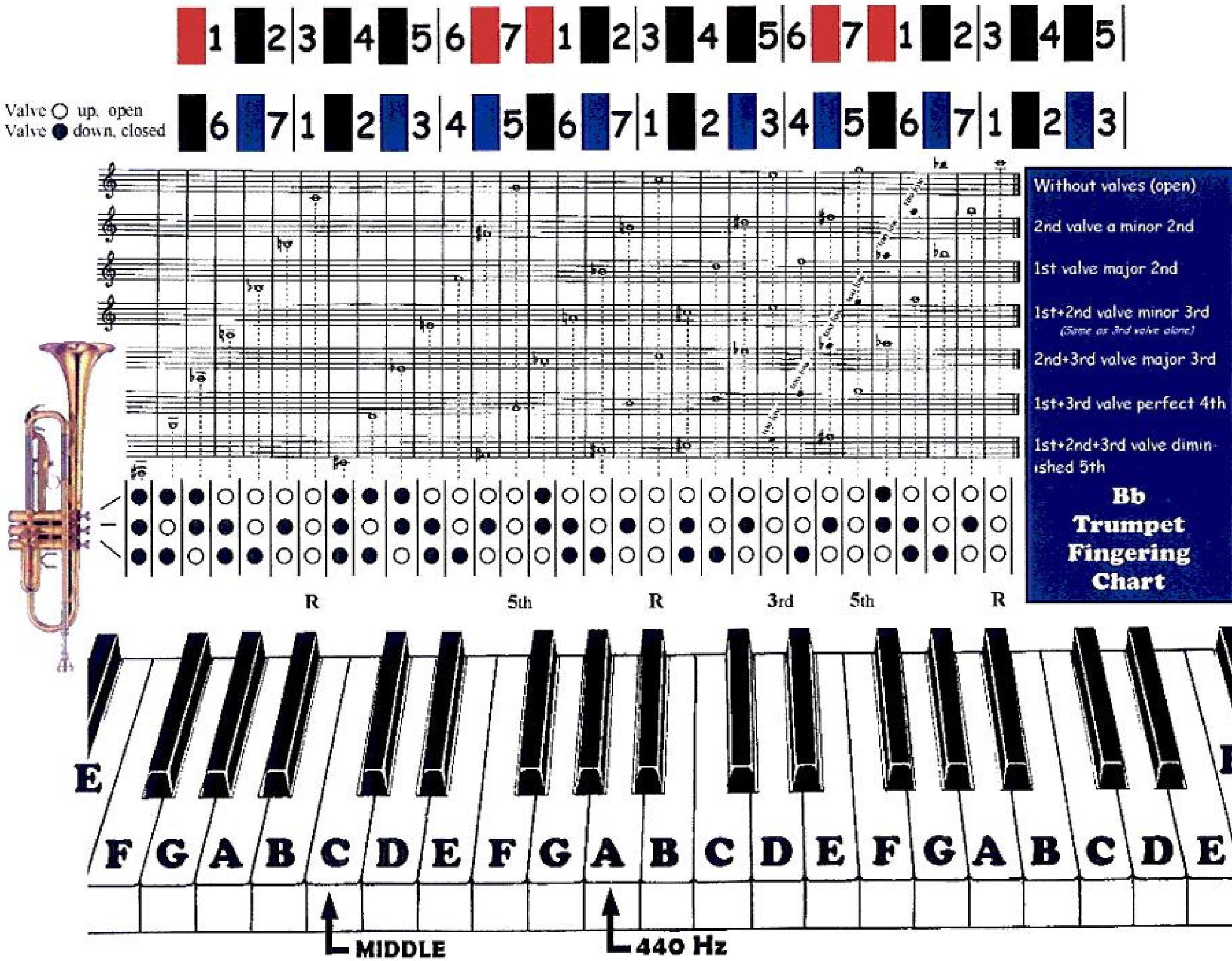


The Victors as a stem plot



(We focus here on the music signal processing...)

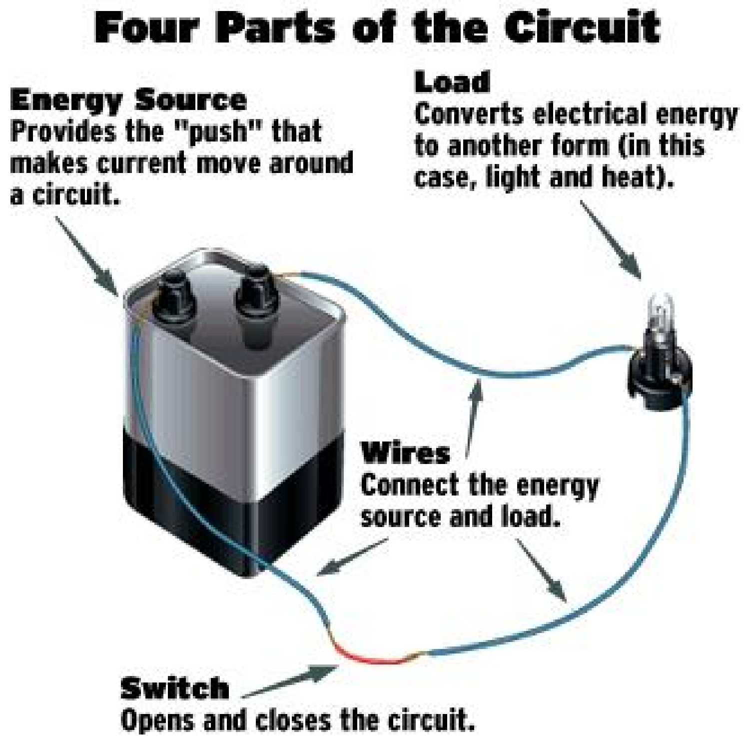
A graphical dictionary



(Citation missing.)

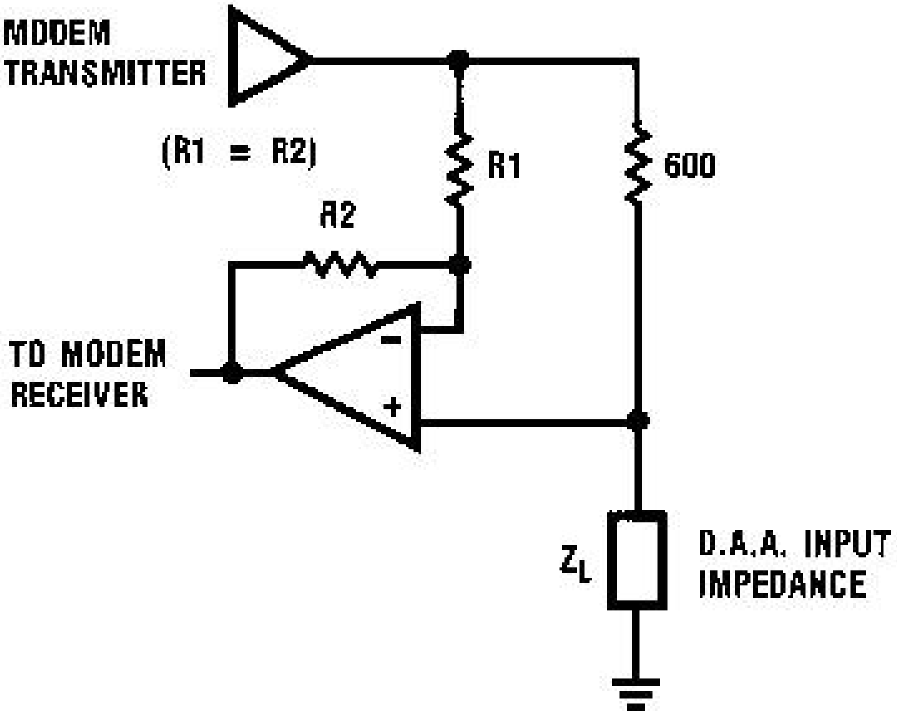
Graphical communication in EE

Picture



http://www.pge.com/microsite/pge-dgz/images/wires/closed-circuit.gif

Diagram with symbols



http://www.epanorama.net/circuits/hybrid-circuit.gif

Frequencies of musical tones

Note	A	A♯,B♭	B	C	C♯,D♭	D	D♯,E♭	E	F	F♯,G♭	G	G♯,A♭
$\frac{\text{FREQ}}{440\,\text{Hz}}$	$2^{0/12}$	$2^{1/12}$	$2^{2/12}$	$2^{3/12}$	$2^{4/12}$	$2^{5/12}$	$2^{6/12}$	$2^{7/12}$	$2^{8/12}$	$2^{9/12}$	$2^{10/12}$	$2^{11/12}$
Hz	440	466.2	493.9	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6
MIDI	69	70	71	72	73	74	75	76	77	78	79	80
Ratio	1/1	16/15	9/8	6/5	5/4	4/3	7/5	3/2	8/5	5/3	16/9	15/8
Just	440	$469.\bar{3}$	495	528	550	$586.\bar{6}$	616	660	704	$733.\bar{3}$	$782.\bar{2}$	825

Frequency ratios are called intervals in music: (Cleve Moler music blog)

- half step (aka semitone) $2^{1/12}/1$
- whole step $9/8 \approx 2^{2/12}$ ratio (2 half steps)
- octave 2/1 frequency ratio (e.g., middle C to next C) (12 half steps): $\left(2^{1/12}\right)^{12} = 2$
- fifth $3/2 \approx 2^{7/12}$ ratio (7 half steps, logically?)

The 3/2 ratio comes from modes of vibrating string (or air column);
used as drones in bagpipes, hurdy gurdy, guitar power chords.

Using the $2^{1/12}$ ratio is called equal temperament (cf. small integer ratios).

J.S. Bach's "Well-Tempered Clavier" (1722)

Some Arabic music uses $2^{1/24}$ ratio (24 notes per octave)

Frequency calculation

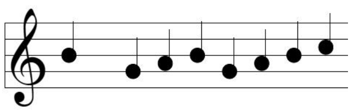
Q0.2 Frequency of the note B that is an octave and a whole step above A440? (Use equal temperament and choose closest.)

A: 880.0 B: 932.2 C: 987.8 D: 1046.5 E: 1108.7 ??

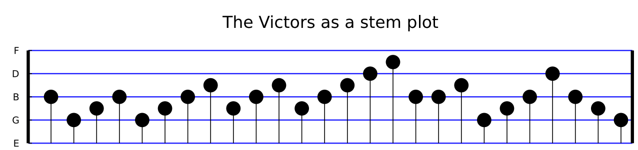
Part 2: Project 1

Project 1: Specification summary

- Pure sinusoidal-tone music synthesizer:
  - `Julia`-generated virtual keyboard via GUI
  - Click on (virtual) key with mouse generates that note
  - Writes audio signal for sequence of played notes into `.mat` file

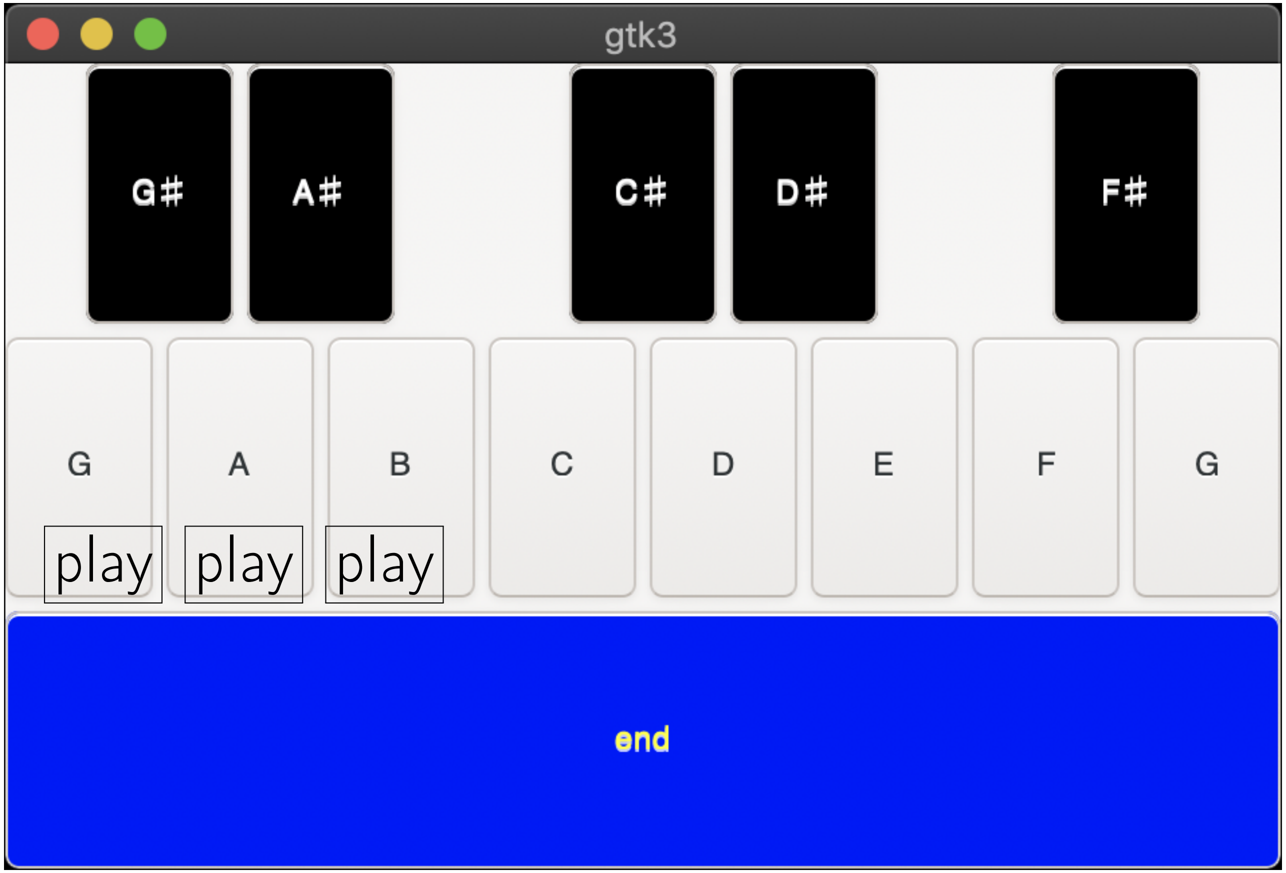

→ user → mouse keypress → Synthesizer: generates sinusoidal tone sequence signal $x[n]$ →  → `file.mat`

- Pure sinusoidal-tone music transcriber:
  - Input: a `.mat` file of music played on synthesizer
  - Output: pseudo-musical-staff notation of music

`file.mat` → Transcriber estimate pitch,duration + display →


- Detailed specifications on subsequent slides.
- Suggestion: output MIDI notation first, then make staff.

Project 1: Tone Synthesizer GUI

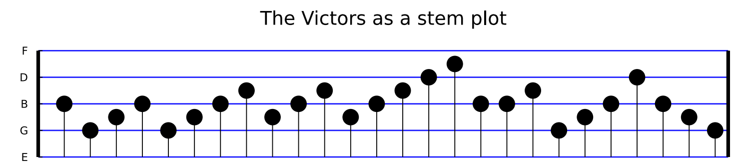


→ signal

- Mimics single octave of a piano keyboard: black (above) and white (below) keys.
- You can jazz this up (add colors, labels) if you so desire.
- Clicking on key with mouse plays that note using `sound`.
- Clicking on "end" causes your song to be played and its signal written to a file using `matwrite`.

Project 1: Tone transcriber output

signal `file.mat` → transcriber: estimate pitch~~, duration~~ + display →



- Crude musical staff notation of "The Victors."
- Musical notes: uses `stem` plot for simplicity.
- No duration information in P1: all notes have same length.
- Heights against 5-line background "staff" must be correct.
- Determining the notes / pitches / frequencies and computing these heights from sampled music signals generated by your synthesizer is the main DSP point of P1.

Julia GUI commands for Project 1

- Laptop human computer interface (HCI) options:
  keyboard, mouse/trackpad, pen/touch, camera, external devices
- Julia has several packages for graphical user interface GUI
  Project 1 & 2 use cross-platform GTK in Gtk.jl (new to me too)
- Template code on Google Drive gives you all the generic GUI stuff;
  you focus on the music signal processing aspects

```
using Gtk
using Sound: sound
using MAT: matwrite

# initialize two global variables used throughout
S = 7999 # sampling rate (samples/second) for this low-fi project
song = Float32[] # initialize "song" as an empty vector

function miditone(midi::Int; nsample::Int = 2000)
    f = 4000 * 2^((midi-96)/11) # compute frequency from midi number - FIXME!
    x = cos.(2pi*(1:nsample)*f/S) # generate sinusoidal tone
    sound(x, S) # play note so that user can hear it immediately
    global song = [song; x] # append note to the (global) song vector
    return nothing
end

# define the white and black keys and their midi numbers - FIXME!
white = ["G" 67; "A" 69; "B" 75; "C" 72; "D" 74; "F" 77; "G" 79]
black = ["G" 67 2; "A" 70 4; "C" 73 8; "D" 74 10; "F" 78 12]

g = GtkGrid() # initialize a grid to hold buttons
set_gtk_property!(g, :row_spacing, 5) # gaps between buttons
```

```
set_gtk_property!(g, :column_spacing, 5)
set_gtk_property!(g, :row_homogeneous, true) # stretch with window resize
set_gtk_property!(g, :column_homogeneous, true)

# define the "style" of the black keys
sharp = GtkCssProvider(data="#wb {color:white; background:black;}")
# FIXME! add a style for the end button

for i in 1:size(white,1) # add the white keys to the grid
    key, midi = white[i,1:2]
    b = GtkButton(key) # make a button for this key
    signal_connect((w) -> miditone(midi), b, "clicked") # callback
    g[(1:2) .+ 2*(i-1), 2] = b # put the button in row 2 of the grid
end
for i in 1:size(black,1) # add the black keys to the grid
    key, midi, start = black[i,1:3]
    b = GtkButton(key * "♯") # to make ♯ symbol, type \sharp then hit <tab>
    push!(GAccessor.style_context(b), GtkStyleProvider(sharp), 600)
    set_gtk_property!(b, :name, "wb") # set "style" of black key
    signal_connect((w) -> miditone(midi), b, "clicked") # callback
    g[start .+ (0:1), 1] = b # put the button in row 1 of the grid
end


function end_button_clicked(w) # callback function for "end" button
    println("The end button")
    sound(song, S) # play the entire song when user clicks "end"
    matwrite("proj1.mat", Dict("song" => song); compress=true) # save song to file
end

ebutton = GtkButton("end") # make an "end" button
g[1:16, 3] = ebutton # fill up entire row 3 of grid - why not?
signal_connect(end_button_clicked, ebutton, "clicked") # callback
# FIXME! set style of the "end" button

win = GtkWindow("gtk3", 400, 300) # 400×300 pixel window for all the buttons
push!(win, g) # put button grid into the window
showall(win); # display the window full of buttons
```

`Julia` concepts and code checks

- layout of GUI via `Gtk...` and `...gtk...`
- callback function (event-driven programming) via `signal_connect`

Q0.3 What is `size(white)` above? Repeated here:

```
white = ["G" 67; "A" 69; "B" 75; "C" 72; "D" 74; "F" 77; "G" 79]
```

A: `(7,2)` B: `(2,7)` C: `(14,)` D: `(14,1)` E: `(1,14)` $\boxed{??}$

Q0.4 What is `length(x)` in `miditone` function above?

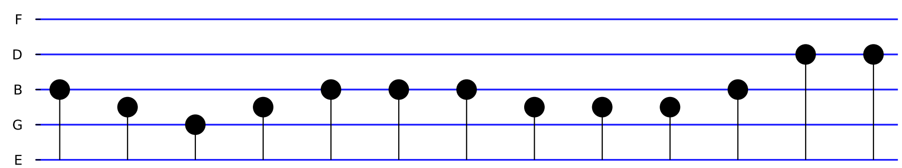
$\boxed{??}$

Julia plotting commands for Project 1

```
using Plots; default(markerstrokecolor=:auto, label="")
y = rand(-0.5:0.5:4.5, 30) # some random "notes" to plot
plot(y, line=:stem, marker=:circle, markersize = 10, color=:black)
plot!(size = (800,200)) # size of plot
plot!(widen=true) # try not to cut off the markers
plot!(xticks = [], ylims = (-0.7,4.7)) # for staff
yticks!(0:4, ["E", "G", "B", "D", "F"]) # helpful labels for staff lines
#plot!(axis=nothing, border=:none) # ignore this
plot!(yforeground_color_grid = :blue) # blue staff, just for fun
plot!(foreground_color_border = :white) # make border "invisible"
plot!(gridlinewidth = 1.5) # try commenting out each of these to see what they do!!!!
plot!(gridalpha = 0.9) # make grid lines more visible
```

- `line=:stem` is key option
- `yticks!` with two arguments puts strings at given locations `0:4`

Example: `y = vec([2 1.5 1 1.5 2 2 2 1.5 1.5 1.5 2 3 3])`



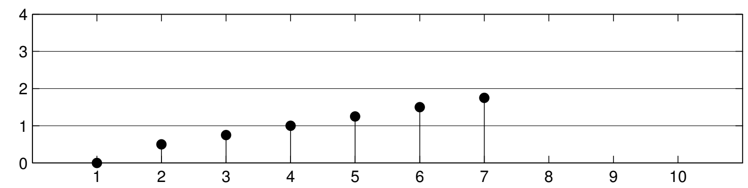
Name that tune ??

Musical tone transcriber: Plotting accidentals

- If using MIDI notation, this would be easy.
  - Find frequencies of each note in Hz; put in vector `F`.
  - Compute MIDI numbers using formula:
    `midi = 69 .+ 12 * log2.(F / 440)`
  - Just stem-plot the result: `plot(midi, line=:stem, ...`
  - But this plot would be poorly suited to musicians
- Challenge: musical staff notation treats natural notes differently from accidentals (sharps & flats)
- Solution:
  - Plot natural notes at 0, 0.5, 1, 1.5, ...
  - Plot accidentals "half way in between" at 0.25 0.75 etc.
  - Use "table lookup" via `Julia` indexing, shown next.

Using a `Julia` vector for "table lookup"

Note	Frequency	MIDI number	vertical position
E		64	0
F		65	0.5
F♯		66	0.75
G		67	1
G♯		68	1.25
A	440	69	1.5
A♯	466	70	1.75
B	494	71	??
C	523	72	??
C♯	554	73	??
⋮			



Q0.5 Position of last '??'

??

Vertical positions of the stem circles are unequally spaced!

Wrong way (works for E, G only): `plot((midi .- 64)/3)`

We need a simple automatic way to specify vertical position given MIDI number.

Solution by table lookup

Define a table (array) of vertical positions (here for E to D only):

```
V = [0 0.5 0.75 1 1.25 1.5 1.75 2 2.5 2.75 3]
```

Use `Julia` indexing to "lookup" vertical positions:

```
plot(V[midi .- 63], line=:stem, ...
```

(discuss indexing, next slide)

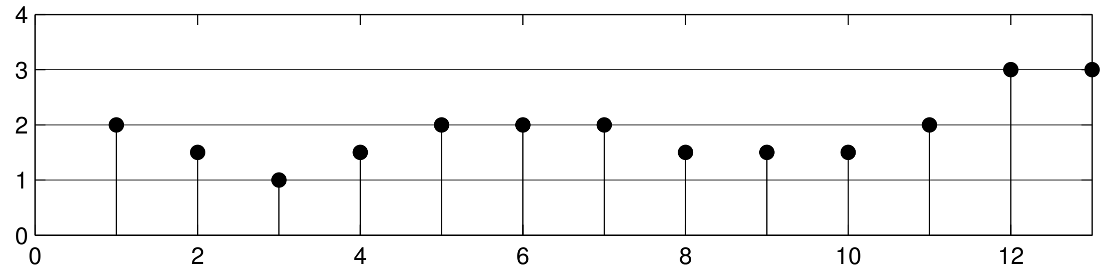
Example (for B A G A B B B A A A ...):

```
midi = vec([71 69 67 69 71 71 71 69 69 69 71 74 74])
```

```
V[midi .- 63]
```

```
2.0 1.5 1.0 1.5 1.5 2.0 2.0 2.0 1.5 1.5 1.5 2.0 3.0 3.0
```

```
plot(V[vec(midi) .- 63], line=:stem, markershape=:circle, ylim=(0,4))
```



Julia array indexing examples

```
x = 10:5:30
```

```
x[2]
15
```

```
x[2:4]
15 20 25
```

```
x[3:end]
20 25 30
```

```
x[[3, 2, 5]]
20, 15, 30
```

```
x[[4; 1:2:3]]
```

$\boxed{??}$

Musical tone transcriber: Outline

- Synthesizer output is a 1D signal with `M` tones concatenated. `x = matread(...)`
- Simplification: transcriber "knows" each tone has length: `N=2000` samples
- Transcriber must determine number of tones `M`. How? ??
- Reshape 1D signal into array of `M` tones of length `N=2000`:
  `y = reshape(x, 2000, :)`

Q0.6 Each tone is one ?? of array `y`.
A: row B: column ??

- Apply simple arccos frequency estimation method:

$$\boxed{f = \frac{S}{2\pi} \arccos\left(\frac{x[n+1]+x[n-1]}{2x[n]}\right).}$$

```
n = 1000; ratio = (y[n+1,:] + y[n-1,:]) ./ 2y[n,:]
F = (S/2pi) * acos.(ratio) # hail to the vectors!
```

(Use Julia to explain about `y[3,:]` etc.)

- Generate MIDI number array from frequency vector `F`.
- Use table lookup to make final stem plot.

Q0.7 If signal `x` has 16000 samples, then `size(ratio)` is:

A: (8,) B: (8,1) C: (1,8) D: 18 E: None of these ??

Project 1: Process

- Create Tone Synthesizer and Tone Transcriber (two jl-files).
  `p1_yourteamname1.jl` `p1_yourteamname2.jl`
  Have your lab instructor confirm that both of these work.
  Upload your two jl-files in a zip file to Canvas.
- Your team will present your project results orally.
  You will learn how to do this in tech comm part.
  Most of the 50 points of your P1 grade based on presentation!
  The rest based on your team's code: does it work correctly?
- Save your presentation as a .pdf file.
  Google slides are convenient for collaborative editing.
  The pdf file name should be `yourteamname.pdf`.
- One team member upload pdf of your presentation to Canvas.
  (Further instructions from discussion leaders.)

DSP/`Julia` in practice - more ways than you realize

juliahub.com/industries/banking-and-finance/

Leverage Julia's superior speed and performance for quantitative finance, trading, optimization, arbitrage, asset management and risk analysis.

Aviva uses Julia to make risk modeling run 1,000x faster

JuliaSIM

Cedar EDA

Why Julia for Finance?

It's simple: Speed + Performance + Scalability + Ease of Use

Aviva, one of the world's largest insurers, is using Julia to comply with the European Union's Solvency II regime. Aviva reports speed increases from 20x up to 1000x faster compared to their legacy system. Furthermore, Aviva reduced the code for compliance models from 14,000 lines of code in a proprietary legacy system to just 1,000 lines of code in Julia. This doesn't just increase speed, efficiency and productivity - it also reduces errors and time spent checking and debugging code.

BlackRock, the world's largest asset manager, uses Julia to power their trademark Aladdin platform.

State Street uses Julia to identify best execution for foreign exchange trading. Conning uses Julia in large scale Monte Carlo simulations for insurance risk assessment.

Other banking, finance and economics users include Atteson Research, the Federal Reserve Bank of New York, Berkery Noyes, Nobel Laureate Thomas J. Sargent, Timeline and Now-Casting Economics.

Interested in Using Large Datasets in Julia?

Watch our webinar to see how to improve the process with JuliaHub.

Register →

Summary for Project 1

- Read Project 1 before lab section this/next week.
- `synth1-template.jl` for GUI code on Google Drive in `dsp-projects` folder

- (return to end of l02 if time permits)

Part 5:
Basic dimension analysis by example

(read yourself if time runs out in class)

Dimensional Analysis Example 1

- Goal: Determine formula for the period of a swinging pendulum, without any physics!
- Find ingredients: mass, length, gravity
- Model: Period = $(\text{mass})^a$ $(\text{length})^b$ g^c
  where g = acceleration of gravity $9.80665 \mathrm{m}/\mathrm{s}^2$
  a, b, c are unknown constants to be found
- Approach: Find exponents using dimensional analysis:

$$\text{time} = (\text{mass})^a \, (\text{length})^b \, (\text{length}/\text{time}^2)^c$$

- No "mass" on LHS so $a = 0$
- No "length" on LHS so $0 = b + c$
- For time: $1 = -2c \implies c = -1/2$, so $b = 1/2$

Model: period = $\text{length}^{1/2} g^{-1/2} = (\text{length}\ /g)^{1/2}$
From physics: period = $2\pi\ (\text{length}\ /g)^{1/2}$
(2π is a unitless constant; cannot be found from dimension analysis.)

Dimensional Analysis Example 2

Prof. Yagle asks:

- If 1.5 people can build 1.5 cars in 1.5 days,
  how many cars can 9 people build in 9 days?
- Do problems like this give you a headache?
- Would you like to solve problems like this with minimal thinking?

Dimensional Analysis Example 2

Prof. Yagle asks:

- If 1.5 people can build 1.5 cars in 1.5 days,
  how many cars can 9 people build in 9 days?
- Do problems like this give you a headache?
- Would you like to solve problems like this with minimal thinking?

Given:
(1.5 cars) / (1.5 days) / (1.5 people) $=$ 2/3 cars / day / people

Now match units:
(2/3 cars / day / people) (9 days) (9 people) $=$ 54 cars

Simply matching the units suffices.

References

[1] M-Z. Poh, N. C. Swenson, and R. W. Picard. A wearable sensor for unobtrusive, long-term assessment of electrodermal activity. IEEE Trans. Biomed. Engin., 57(5):1243–52, May 2010.