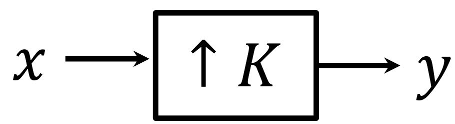
ECE 420
Lecture 4
Feb 11 2019

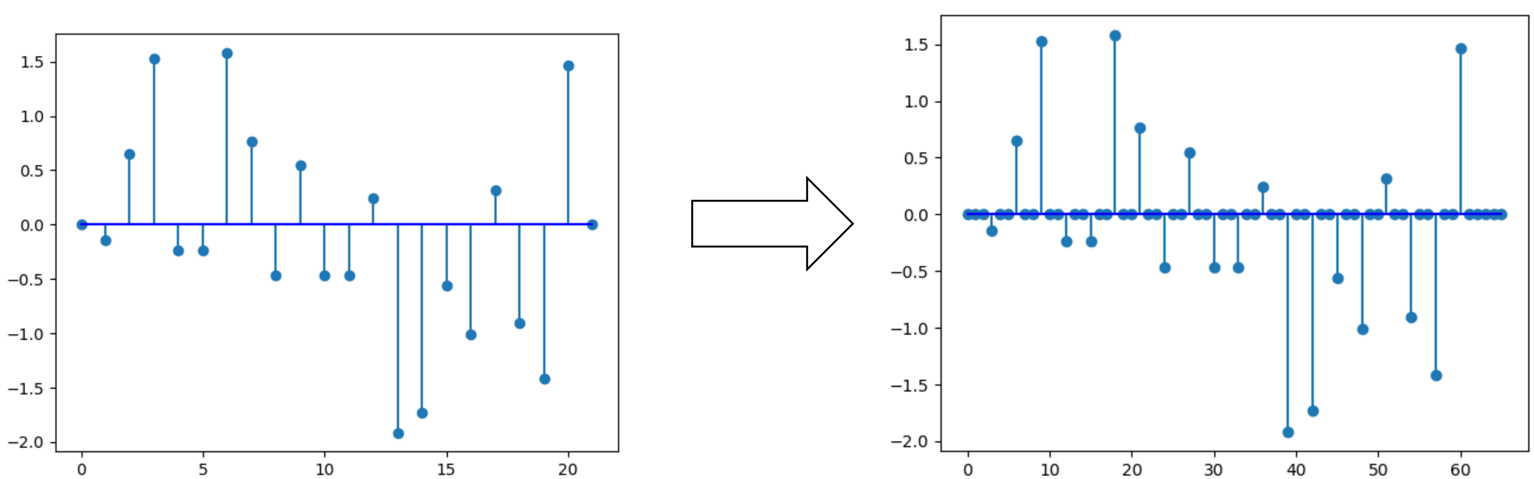
Signal Resampling

- General problem statement:
  - We have samples of a signal
  - These aren’t the sample positions we want
    - Different sampling rate
    - Different sampling locations
    - Both of the above
- What algorithms can we use to synthesize the signal at the desired output locations?

Rate Changing Operators - Upsampler

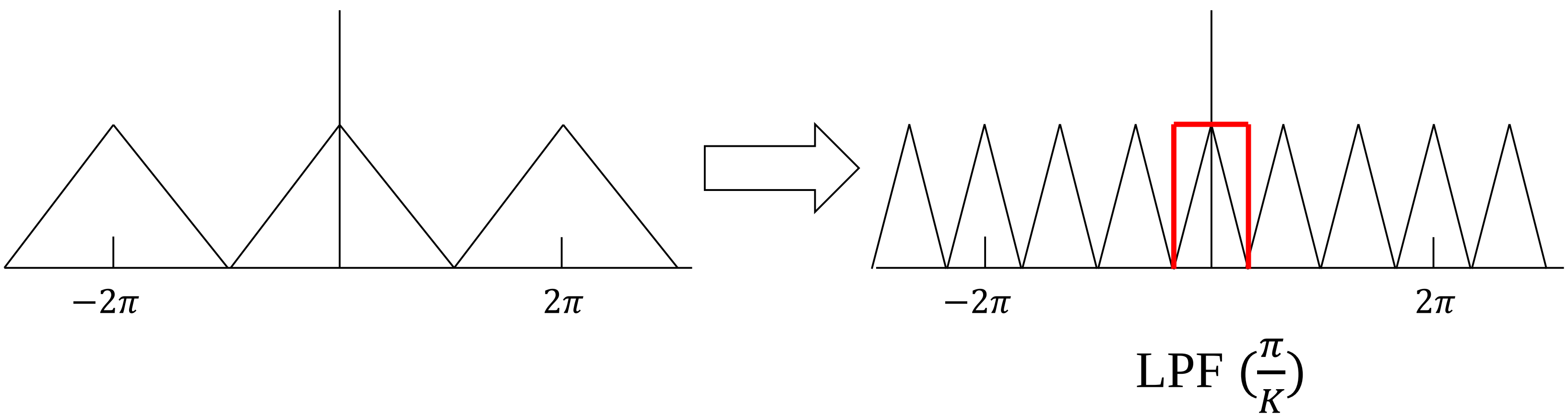
- Performs zero insertion on the signal
  - Add K-1 zeros between each sample
- Always ‘safe’ as we do not lose any data





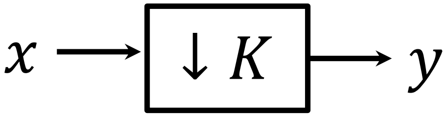
Rate Changing Operators - Upsampler

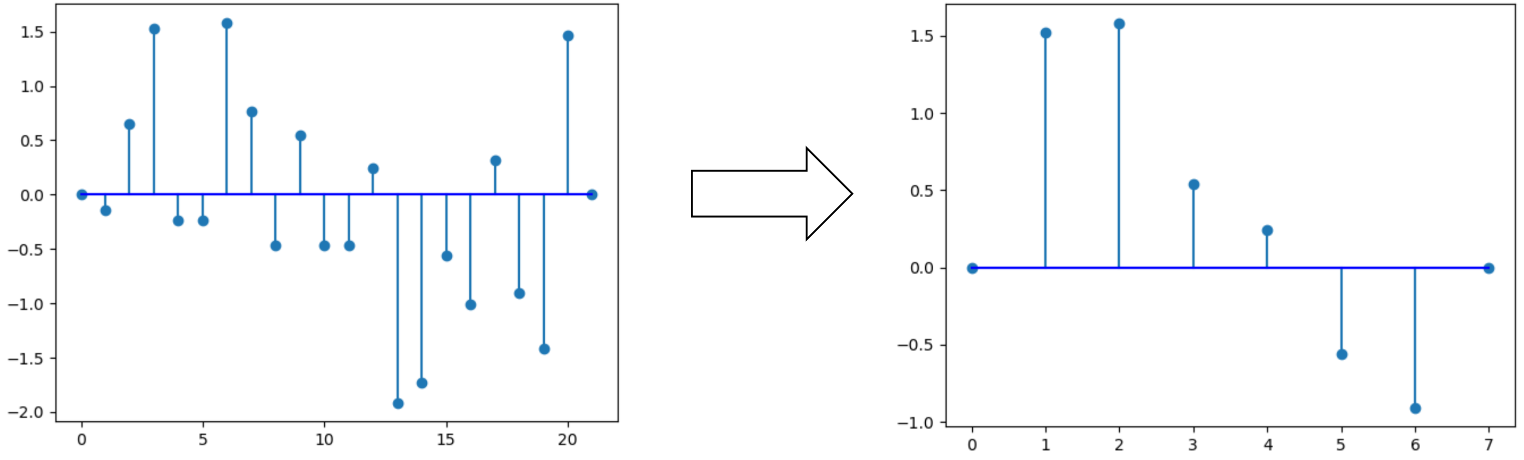
- "Compresses" spectrum by a factor of K
  - $Y(\omega) = X(\omega K)$
- Introduces aliased copies
- How can we eliminate aliased spectra?



Rate Changing Operators - Downsampler

- Reduce the number of samples in the signal
  - Keep first sample out of every batch of K samples
- Potentially unsafe as we are discarding samples



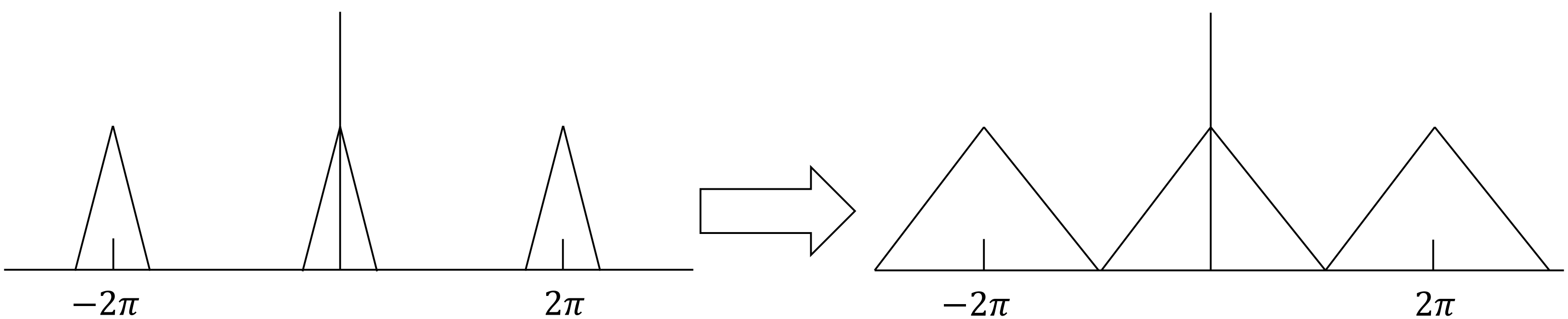


Rate Changing Operators - Downsampler

- "Expands" spectrum by a factor of K

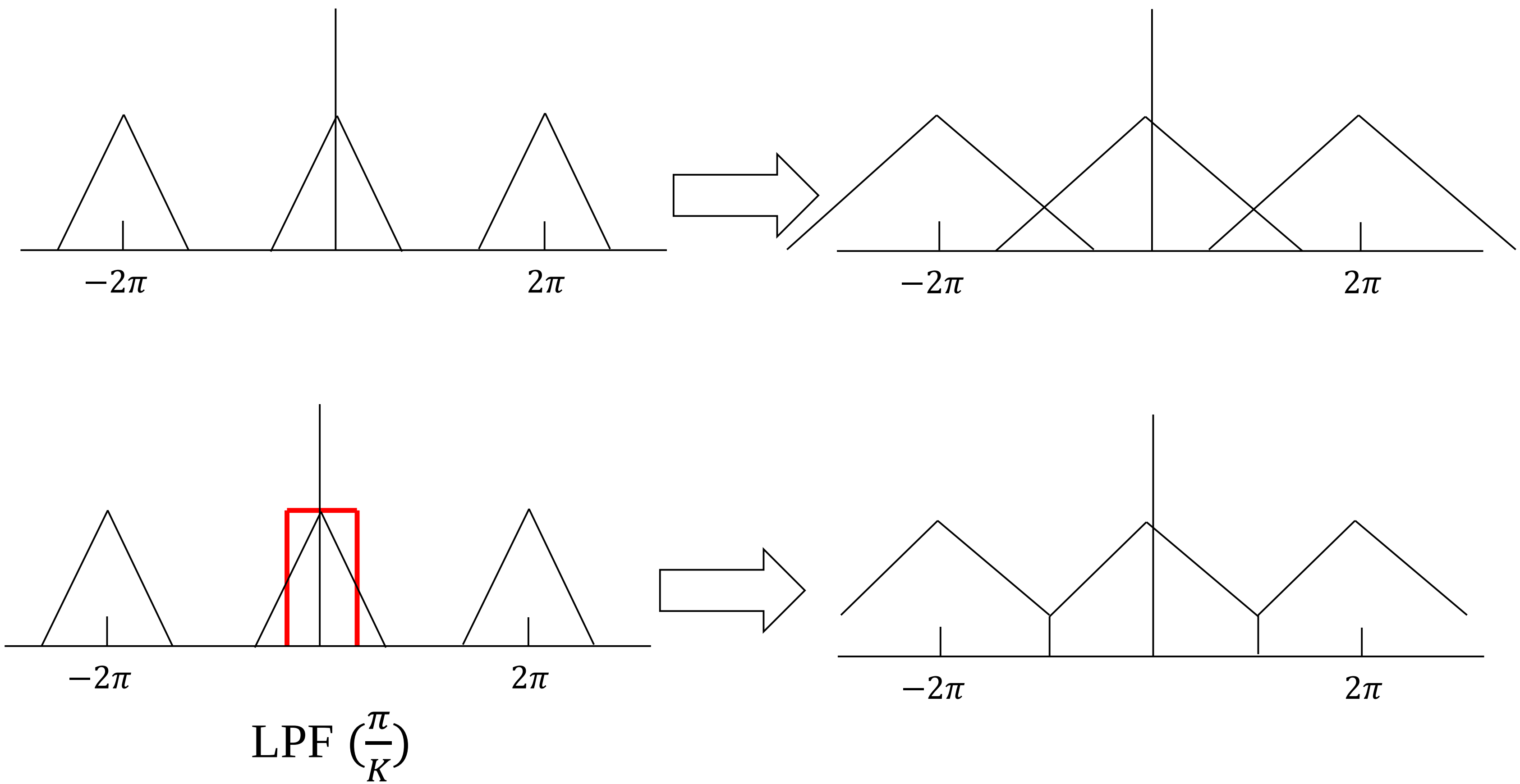
$$Y(\omega) = \frac{1}{K}\sum_{k=0}^{K-1} X(\frac{\omega - 2\pi k}{K})$$

- Potential for aliasing to occur. Caution!



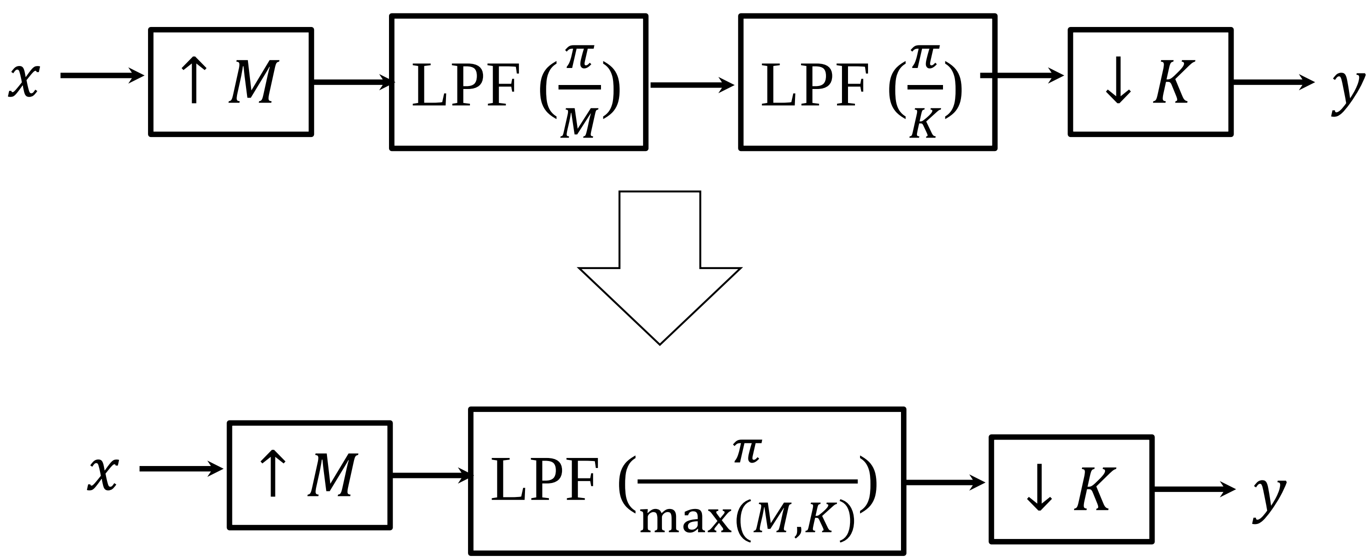
Rate Changing Operators - Downsampler

- How can we prevent aliased spectra?



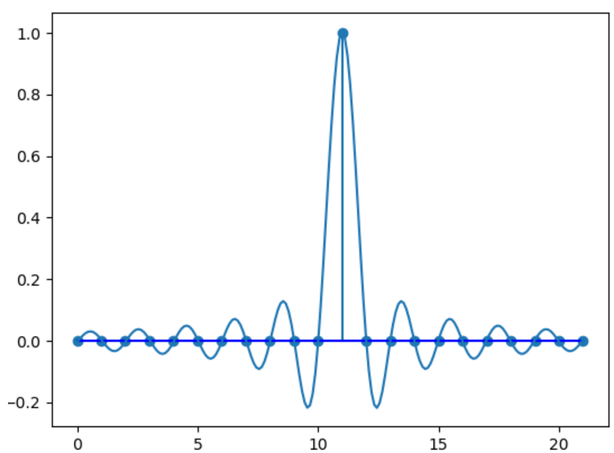
Rate Changing - Fractional Rates

- Upsampling/downsampling operations defined for integer K
- How can you implement arbitrary fractional rates?
  - Cascade of Upsampler (rate M) followed by Downsampler (rate K)
  - Effective rate change of M/K
  - Why upsampling first?

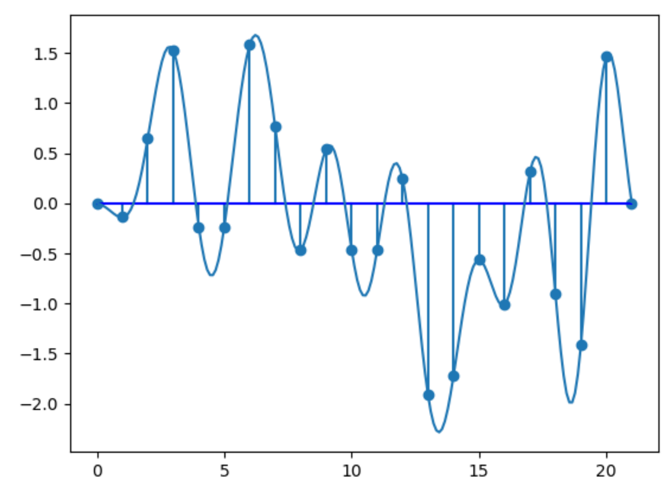


Upsampling as Interpolation

- Another interpretation of upsampling with an LPF is an interpolation operation
- Interpolation kernel is the impulse response of the LPF
- Interpolated signal is this IR centered at each upsampled sample position and added together



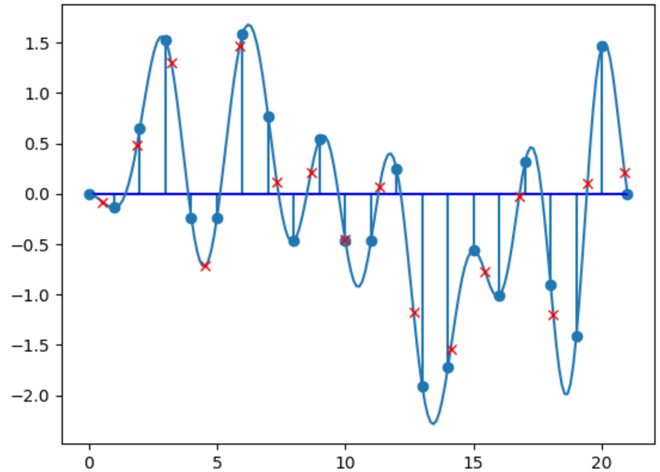
LPF Impulse Response



Interpolated Signal

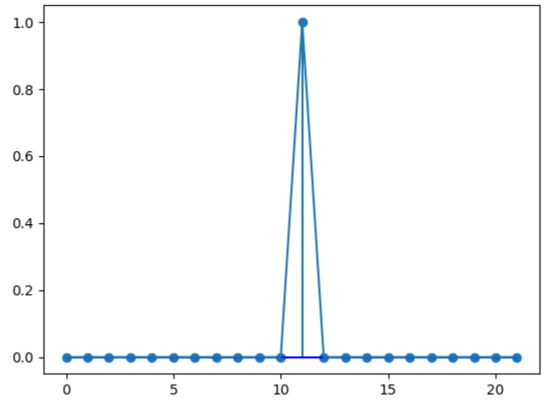
Direct Interpolation

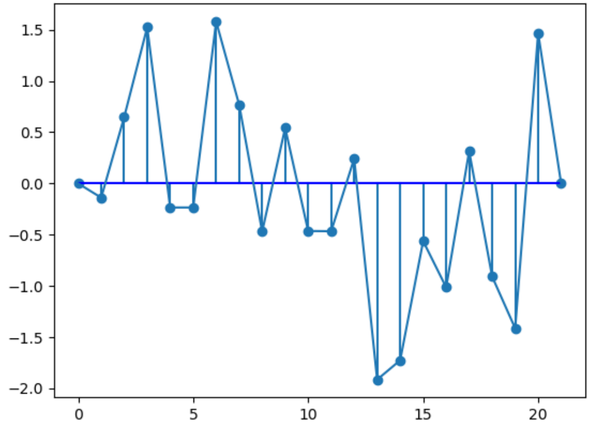
- Efficient filtering works for integer upsampling due to consistency of relative offset of desired sample locations to input sample locations
- For rational rate changes this is not the case
- We can still use the interpolation interpretation to directly resample the signal at arbitrary positions
- Can be costly due to large support of interpolation kernel



Alternate interpolation basis

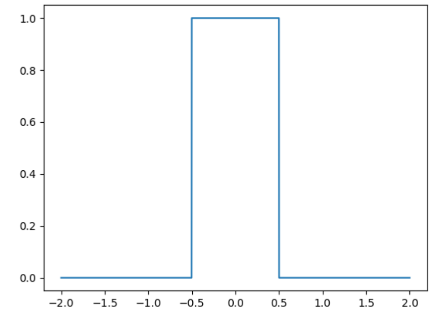
- Generally speaking, recast the problem as a D-to-A-to-D
  - $x(t) = \sum x[k]\phi(t-k)$
- Therefore we can resample at an arbitrary position of x by evaluating x(t) at the desired non-integer t positions
  - $y[n] = x(\tau_n) = \sum x[k]\phi(\tau_n - k)$
- Assuming $\phi(t)$ has small support, only a small number of samples in $x[n]$ are required
- Linear interpolation (or 'tent function') is one such option



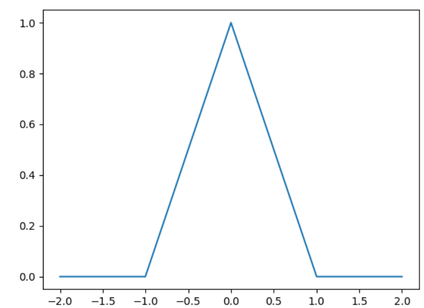


Spline Interpolation Basis

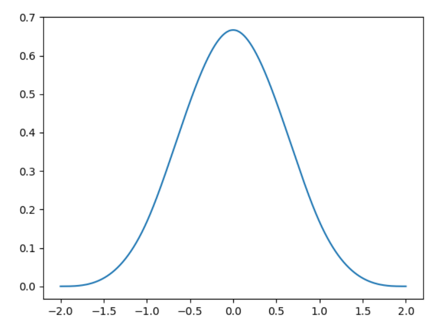
- Splines are recursively defined
  - $\beta_0(t) = \begin{cases} 1 & |t| \leq \frac{1}{2} \\ 0 & |t| > \frac{1}{2} \end{cases}$
  - $\beta_n(t) = \beta_{n-1}(t) * \beta_0(t)$
- β_0 is a box, β_1 is a tent, higher orders are progressively smoother, with more regularity
- Can provide good interpolation performance at reasonable computational cost
- Higher order splines no longer an interpolating function
  - Must perform a spline transform first



β_0



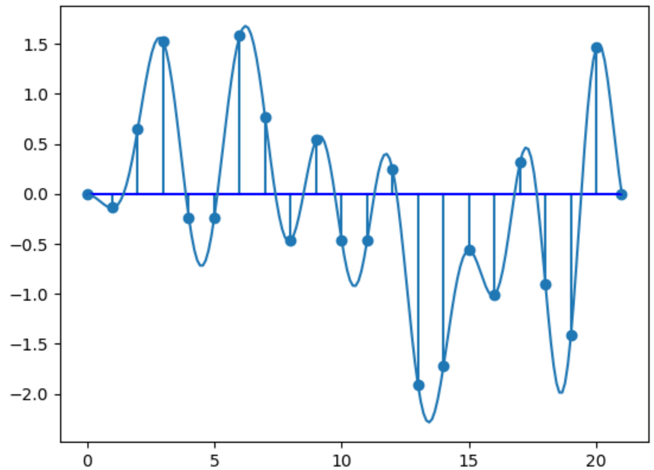
β_1



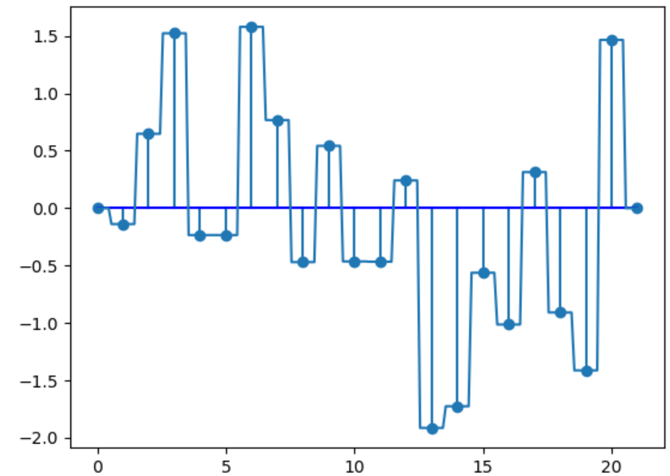
β_3

Interpolation Comparison

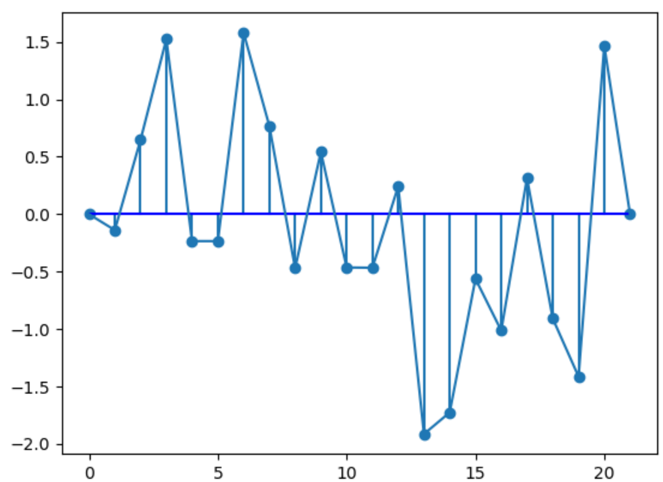
Sinc/LPF



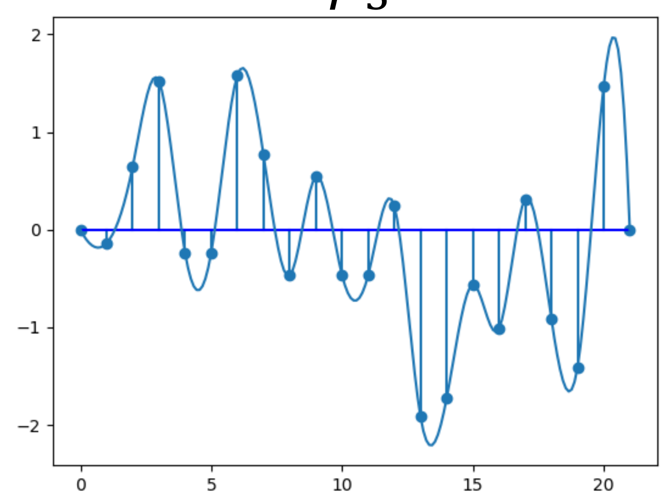
β_0



β_1



β_3

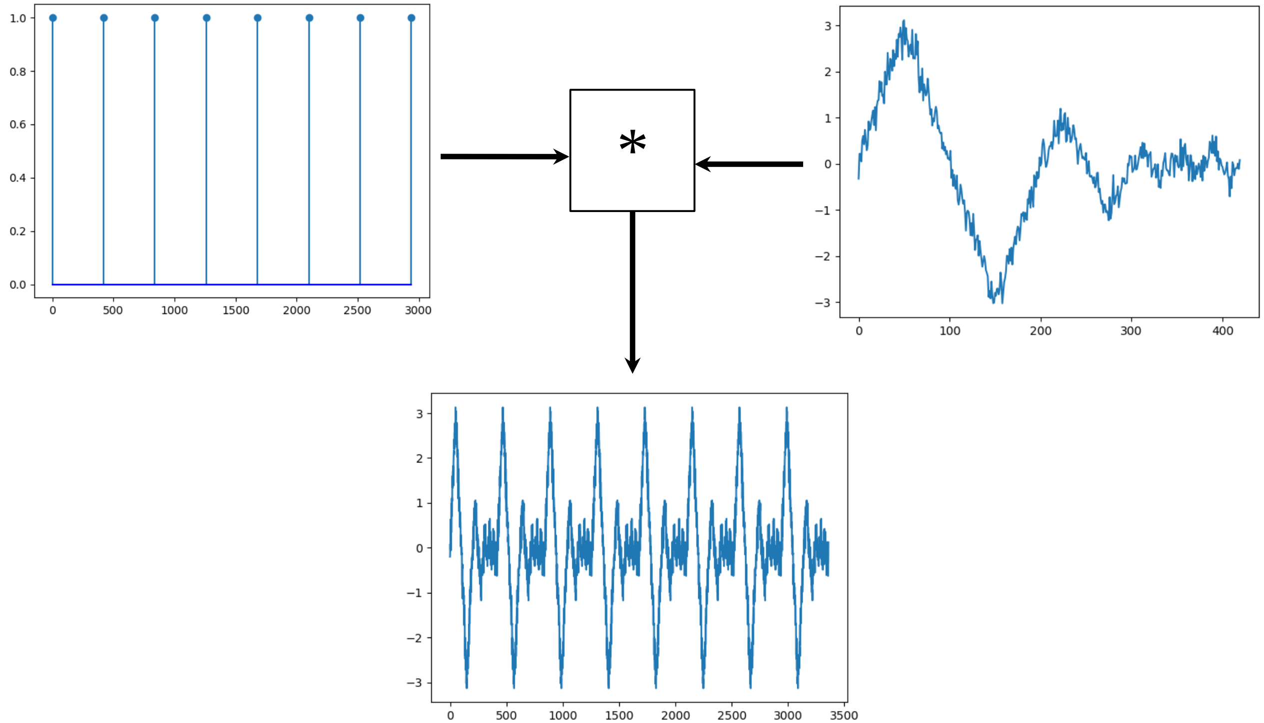


TD-PSOLA

- TD - Time domain
  - Operating directly on the signal samples, no domain transformation
- PS - Pitch Synchronous
  - Operations revolve around reference points (epoch markers or pitch-marks) corresponding to the input or desired pitch of the signal
- OLA - Overlap-Add
  - The synthesized signal is produced by signals positioned about the pitch-marks, where those signals overlap and are added together to form the final output
- Prosody/Prosodic – Patterns of stress and intonation in a language

Speech Synthesis Model

- Speech production initiated as a pulse train
- Vocal tract / mouth / tongue / etc. create a transfer function
- Spoken voice is a 'convolution' of these functions



Pitch-synchronous Manipulation

- Let production model be $y[n] = x[n] * h[n]$
- Spacing of the pulses/delta functions defines pitch of the signal
- Main idea behind pitch synchronous processing is to
  - Identify delta locations of $x[n]$ and filter $h[n]$
  - Manipulate the delta locations to alter the signal to have the desired characteristics
  - Resynthesize the modified signal by reapplying $h[n]$
- For example, if we want to change the pitch to a new value $\hat{P}$
  - $\hat{x}[n] = \sum \delta\left[n - \hat{P}k\right]$
  - $\hat{y}[n]$= $\hat{x}[n] * h[n] = \sum h\left[n - \hat{P}k\right]$

Types of Pitch-Synchronous Modifications

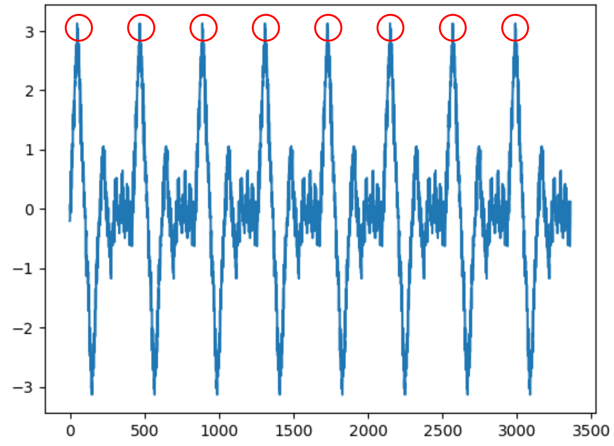
- Pitch-scale modifications
  - Modify the pitch of the signal to a desired target pitch
  - Focus of the lab
- Time-scale modifications
  - Modify the time extent of the signal without changing the pitch
- Pitch- and time- scale modifications
  - Can be done as a cascade of operations or jointly
  - Jointly allows us to skip reprocessing of the signal and manipulate the delta positions in one step

Challenges

- Varying pitch over time
  - Even variation with a ‘constant’ pitch region
- Variations in speech waveform over time
  - Uniform $h[n]$ assumption does not hold
- Preventing distortions in the synthesized signal
- Block processing of the audio frames

Epoch Definition

- We want to extract the delta positions
- While mostly regular, they do not follow an exact spacing
- Also $h[n]$ varies with time
- Attempt to pick a consistent point within each $h[n]$
  - Denote this the epoch or pitch-mark
  - Strategy: search for the maximum value within each estimated pitch interval

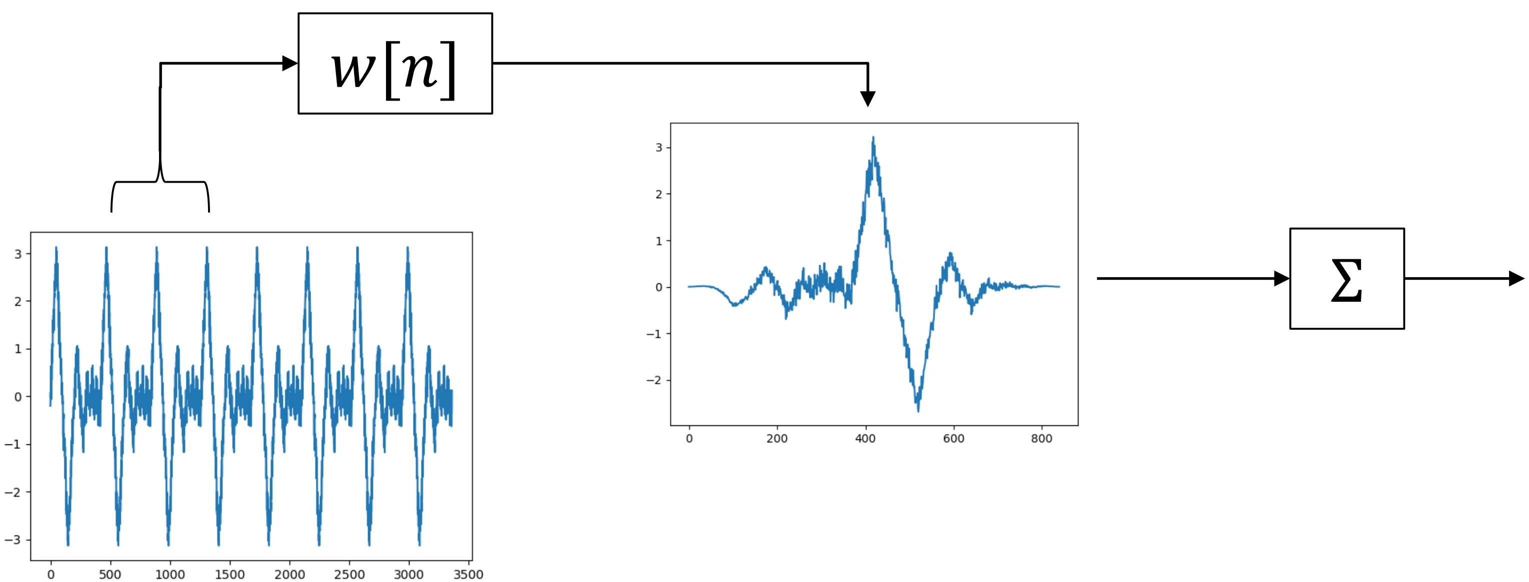


Epoch Mapping

- Establish relationship between input and output epoch points
- Input epochs: from pitch / waveform analysis
- Output epochs: regularly spaced positions at target pitch / time duration
- Algorithm: For each output epoch location, find the nearest input epoch location

Signal Synthesis

- To accommodate variations in $h[n]$, estimate over two adjacent periods
- Window $h[n]$ to taper transitions
- Position at output epoch points
- Combine all outputs together to form synthesized signal

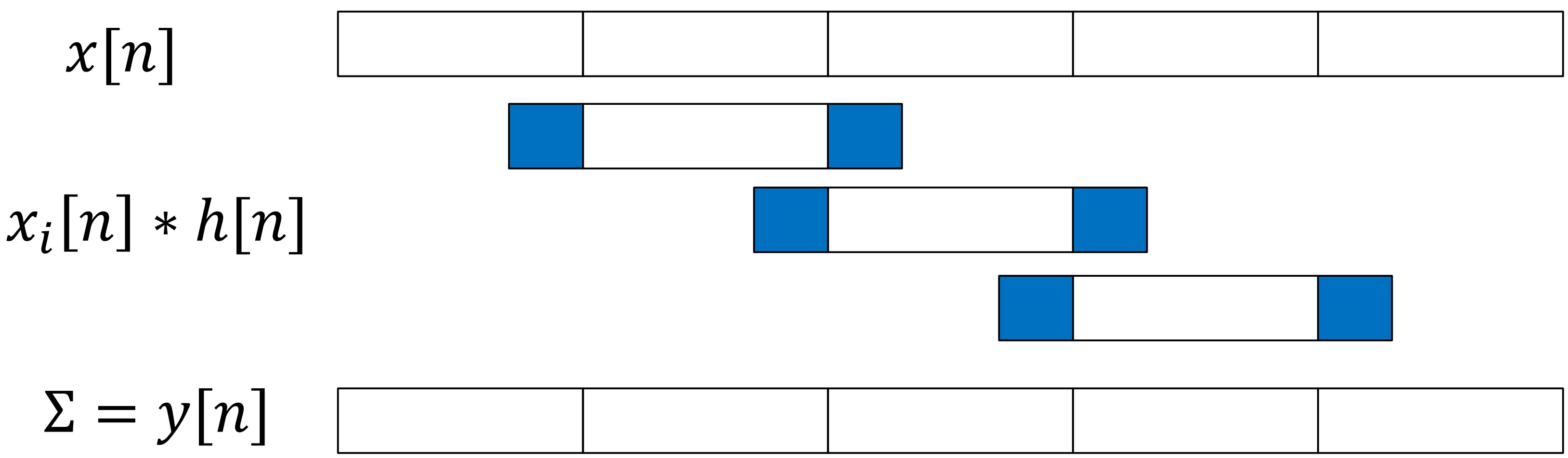


Block Processing Challenges

- Data is broken up into blocks/frames of data for processing due to practical reasons
  - Memory
  - Responsiveness
- Depending on the algorithm, there may be dependencies among blocks of data
- How can we address this problem?
  - Buffering!
- Be aware of impact on
  - Memory Footprint
  - Latency

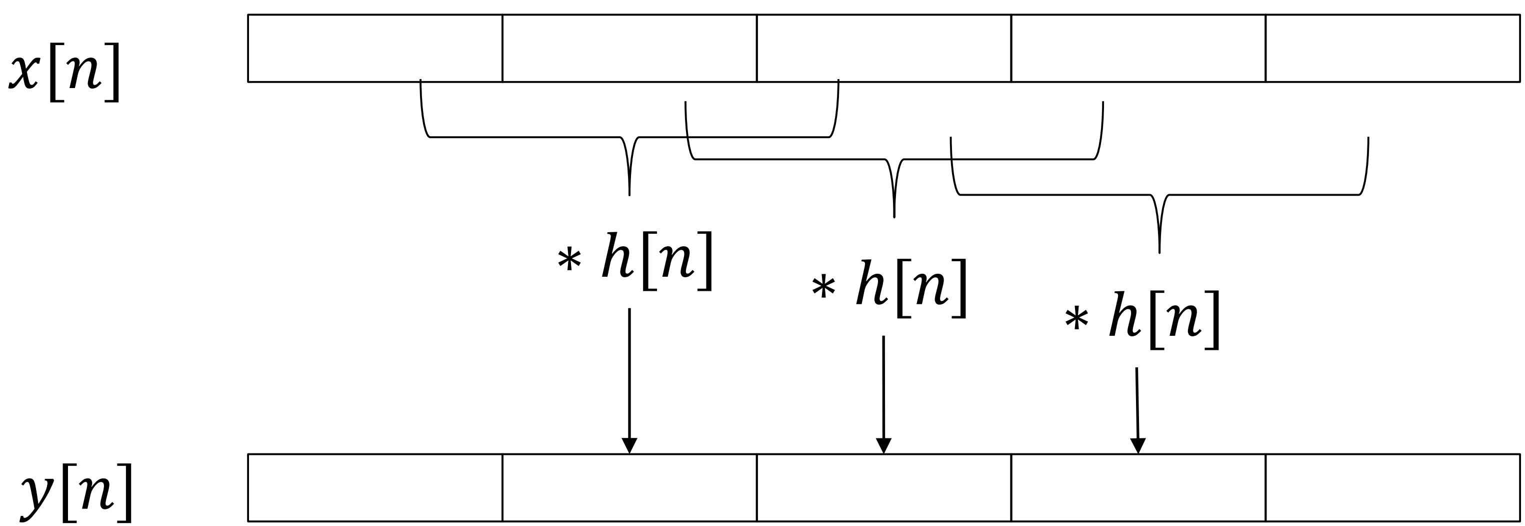
Overlap-Add

- Consider a filtering operation application
- Application of the filter to a frame of data will result in an output wider than the input frame
- Buffer this output in a larger 'working' output buffer and aggregate block outputs
- Send off an output block once all contributions are complete



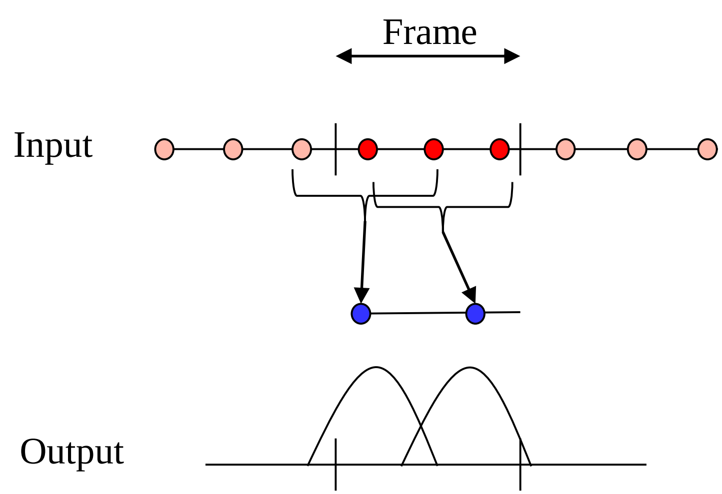
Overlap-Save

- As opposed to buffering outputs we can instead buffer inputs
- 'Work backwards' from a given output block to determine what input data is required to produce it
- Buffer all input data that falls outside of block boundaries



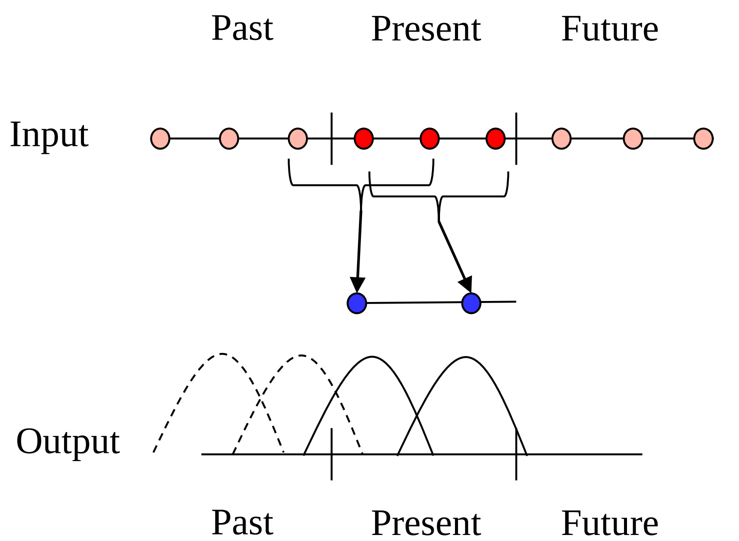
PSOLA Block Processing

- Two main issues that arise from framing the data
  - Depending on epochs selected, windowed interval may stretch across multiple input frames
  - After repositioning on output epoch location, windowed response may stretch across multiple output frames

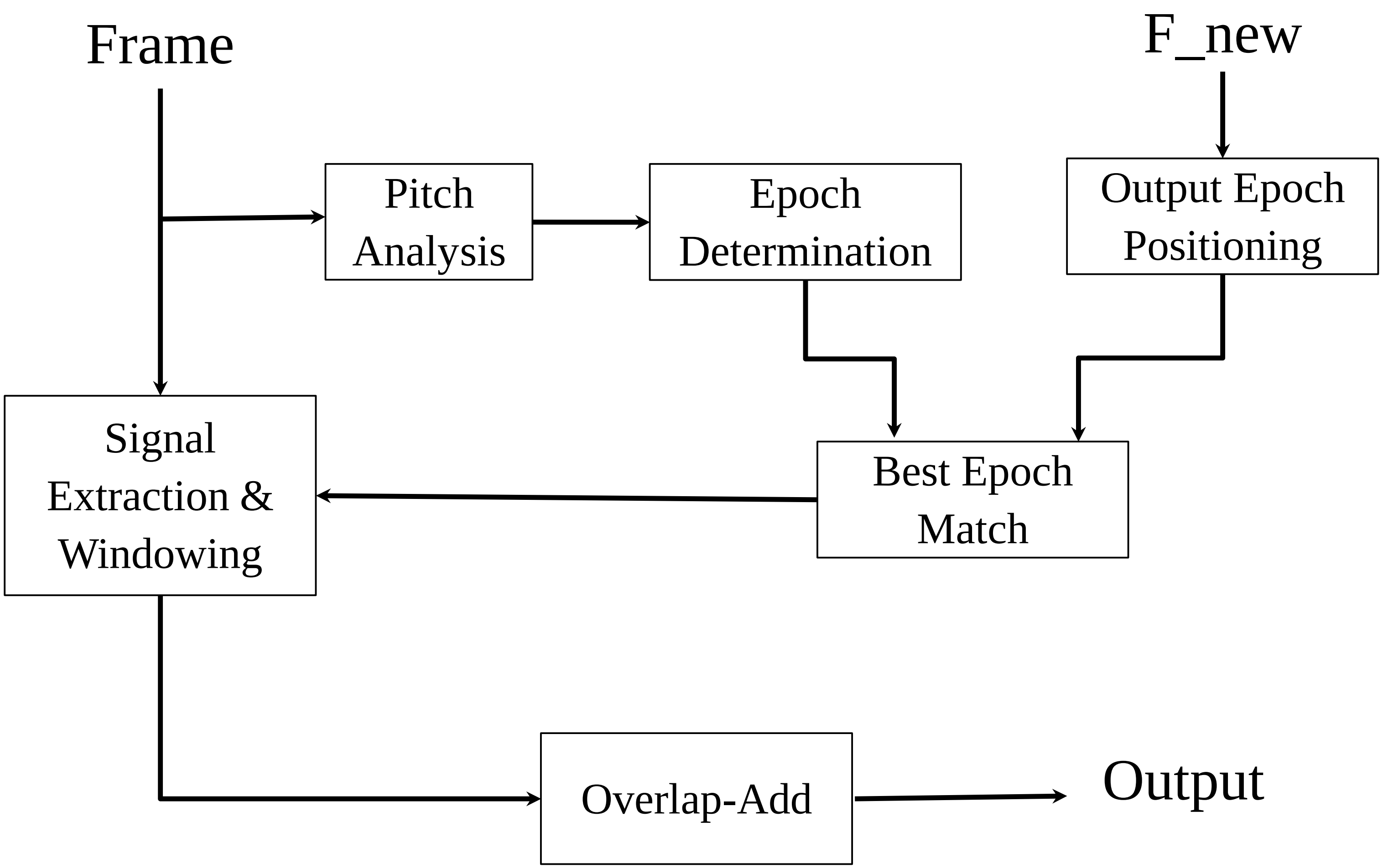


PSOLA Block Processing

- Approach: Keep buffer of input blocks and output blocks
  - Denoted ‘past’, ‘present’, and ‘future’
- Determine contributions for output epoch points in the ‘Present’
- Allow impulse response to spill over into ‘Past’ and ‘Future’
- After all ‘Present’ points processed ‘Past’ will be complete, ready to emit
- Shift down Present to Past and Future to Present



Pitch Synthesis Algorithm Summary



PSOLA Variations

- Linear Prediction PSOLA
  - Effectively tries to model $h[n]$ and decompose speech to find excitation signal $x[n]$
  - Manipulate $x[n]$ as desired and then reapply filter to synthesize speech
- Fourier-Domain PSOLA
  - Perform STFT on pitch periods
  - Estimate spectral envelope of speech and divide out
  - Modify pitch harmonics
  - Reapply spectral envelope and inverse STFT

This week

- Lab 4: Pitch Analyzer Quiz/Demo
- Lab 5: Pitch Synthesizer
  - Linked video recommended!
- Be thinking about Assigned Project Labs / Groups
  - Proposal due March 1