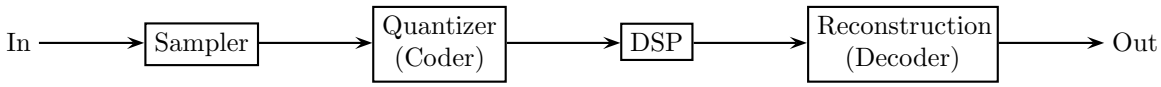
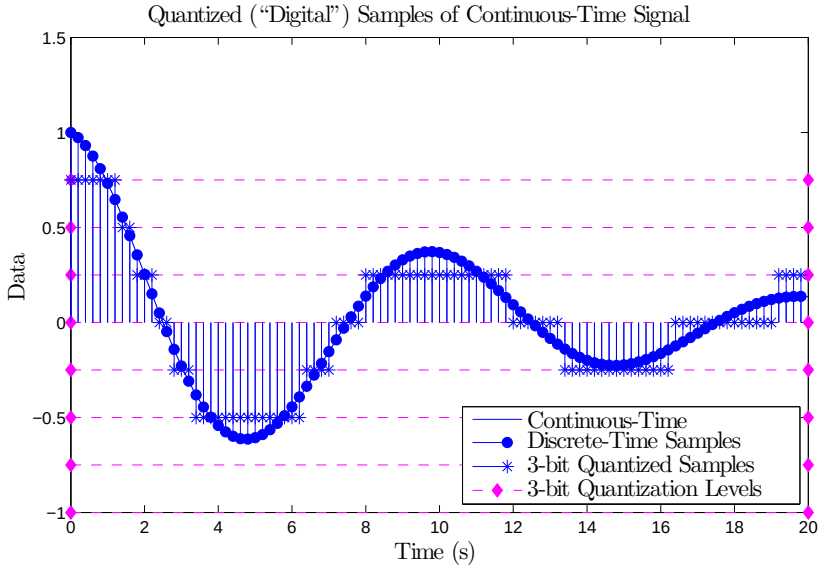
ECE 557: *Control, Signals, and Systems Laboratory*

Notes for Lab 2 (Introduction to Digital Signal Processing)

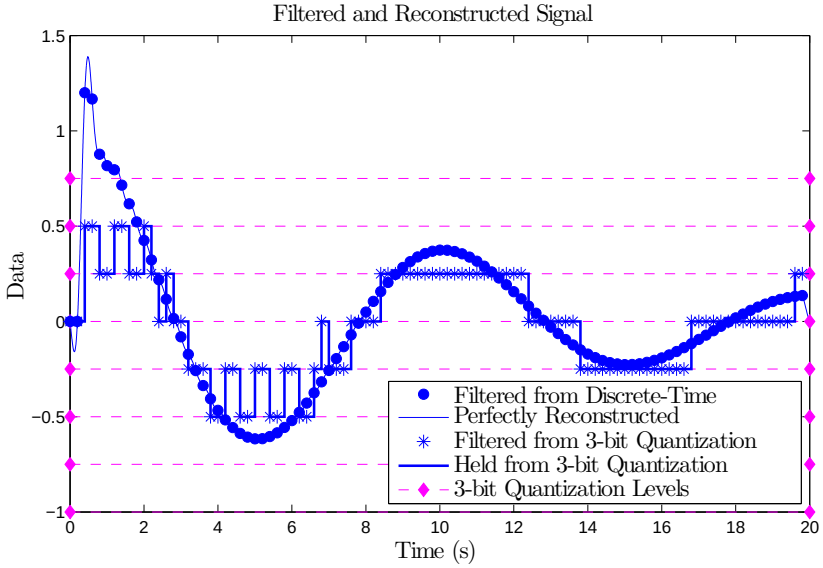
1. Overview: Sampling, Quantization, Filtering, Reconstruction



The signal $x(t) = e^{-0.1t}\cos(2\pi 0.1t)$ is sampled at 5 Hz. The resulting samples are $x[k] \triangleq x(k/5) = e^{-0.02k}\cos(2\pi 0.02k)$. Those samples are quantized for processing by a 3-bit DSP.



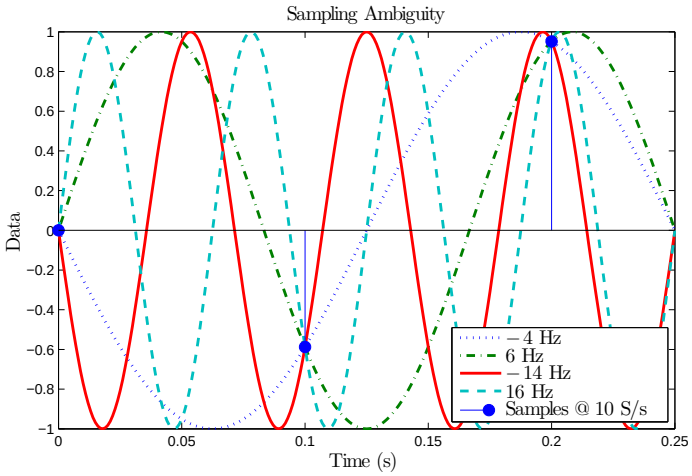
Ideally, the discrete-time signal is filtered so that $y[k] = 1.2x[k-2] - 0.2y[k-2]$, but due to quantization for *every* operation, the output is less than ideal.



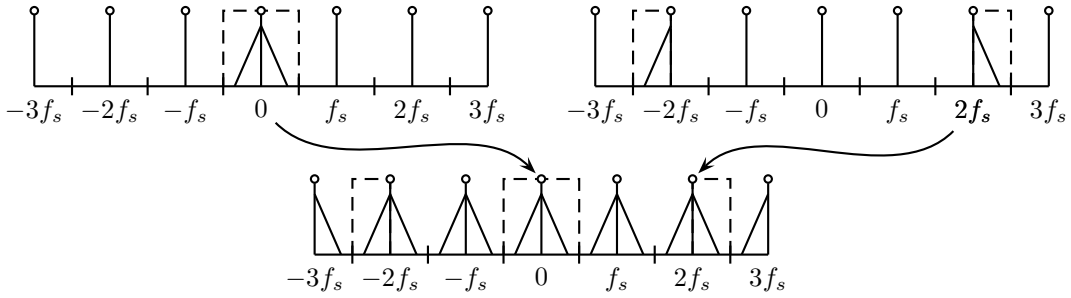
(Here, "perfect" reconstruction uses sinc interpolation.*)*

2. What happens when sampling every T seconds?

   - Sampling multiplies a time-domain signal by a pulse train.
     - Multiplication in the time domain is convolution in the frequency domain.
     - A pulse train separated by T s in the time-domain corresponds to a pulse train separated by $1/T$ Hz in the frequency domain.
     - Convolution by an impulse corresponds to shifting.
     - So sampling *copies* a baseband signal to an ∞ windows centered at $1/T$ Hz multiples.
   - So sampling a signal creates *ambiguity*. Notice how samples intersect multiple sine waves.

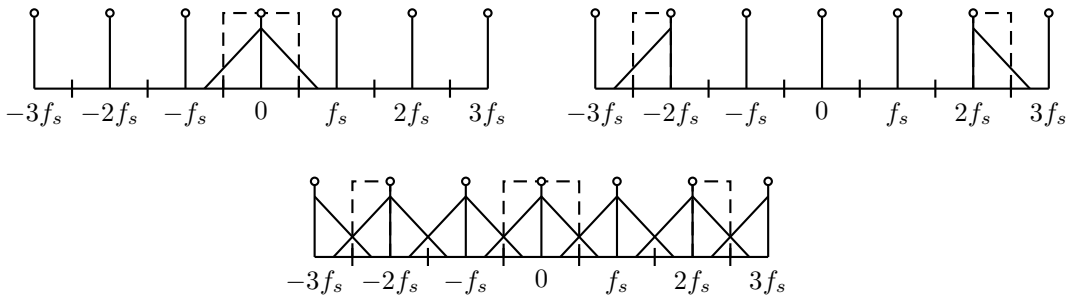


   - To properly reconstruct, apply a bandpass filter to the half window you want.



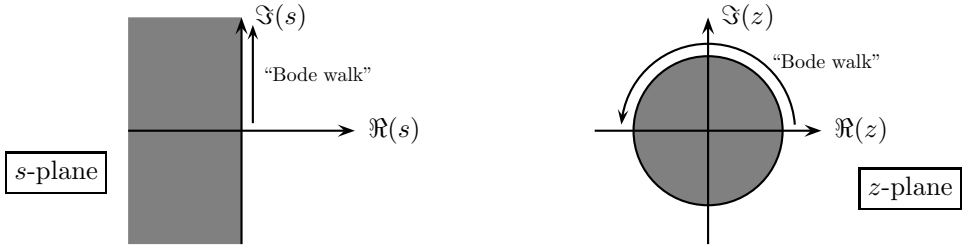
     Consider snapshots of a wagon wheel (e.g., from a camera or jittering of the eye). They can be reconstructed forward and backwards at different speeds. Our eyes and brain may automatically choose the baseband image, but with concentration, the other images are possible.

   - The *band-limited* signal must fit within a *single half window* — it cannot *overlap* two half windows (*Shannon–Nyquist Sampling Theorem*). Constraints are independent of the "highest frequency."



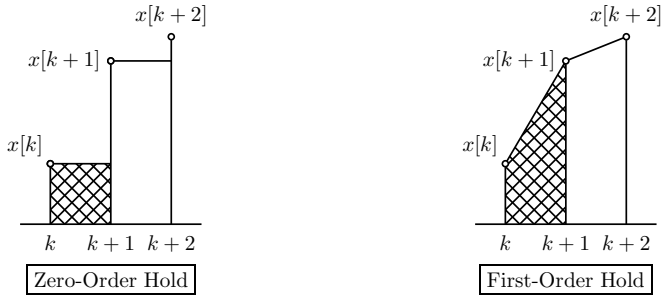
3. The old s-domain and the new z-domain.

- In continuous time, signals can be built out of sums (or integrals) of terms of the form Ae^{st} where s is a complex number of the form $\sigma + j\omega$.
  - For example, the damped sinusoid $2e^{-5t}\cos(\omega t) = e^{(-5+j\omega)t} + e^{(-5-j\omega)t} = e^{-5t}(e^{j\omega t} + e^{-j\omega t})$.
  - Linear-time-invariant (LTI) differential equations have the special property that lets us analyze individual components of signals *separately* and add them later.
  - Conveniently, differentiating functions like Ae^{st} is *equivalent* to scaling them by s, and $e^{st} \neq 0$.
  - Consider the LTI equation $y' + y = x'$.
    - Solving for y when $y' + y = 0$ gives us the behavior for no input.
      - Let $y(t) = Ae^{st}$, which means $s + 1 = 0$.
      - So $y(t) = e^{-t}$ is the solution, and because $s < 0$, it eventually dies out.
      - Because the natural response of the signal *dies out*, the system is ***stable***.
    - Similarly, input signals that have $x' = 0$ have no impact on the system output.
      - Let $x(t) = Ae^{st}$ and solve for s.
      - As expected, solution is $s = 0$ (i.e., $x(t)$ is constant), and so constant inputs are ignored.
      - These are the "zero dynamics" of the system.
    - So long as the system is stable, we can assume $y(t) = Be^{st}$ and $x(t) = Ae^{st}$ and solve for *transfer function* B/A to get $s/(s+1)$.
      - Using the complex plane as the floor, the transfer function's *magnitude* is like a *tent*.
      - The tent is held up by its "poles" and pinned down by its "zeros."
      - An LTI system can be completely characterized by sampling its response to a sinusoids of varying frequencies. This data can be put into a *Bode* plot.
      - The Bode plot is the shape of the tent along the imaginary axis from DC to ∞.
- In discrete time, signals are sampled every T time so that $x[k] \triangleq x(kT)$.
  - So substitute $t = kT$ into the prototypical Ae^{st} to get Ae^{sTk} or Az^k where $z \triangleq e^{sT}$.
  - Conveniently, iterating from $x[k]$ to $x[k+1]$ is equivalent to scaling by z.
    - If $x[k] = Az^k$, then $x[k+1] = Az^{k+1} = Az^k z = z\,x[k]$.
  - Consider the LTI *difference equation* $y[k+1] - 0.5y[k] = x[k+1] - x[k]$.
    - Solve for $y[k]$ when there is no input.
      - Let $y[k] = Az^k$, which means $z - 0.5 = 0$.
      - So $y[k] = 0.5^k$ is the solution, and because $|z| < 1$, it eventually dies out.
      - Because the natural response dies out, the system is ***stable***.
    - Similarly, input signals with $x[k+1] - x[k] = 0$ have no impact on system output.
      - Letting $x[k] = Az^k$, we find $z = 1$.
      - So all constant signals are ignored.
      - These are the "zero dynamics" of the system.
    - Because the system is stable, let $y[k] = Bz^k$ and $x[k] = Az^k$.
      - *Transfer function* B/A is $(z-1)/(z-0.5)$.
      - Again, use the *tent analogy* with "poles" pulling up and "zeros" pulling down.
      - Here, the Bode plot traces the **CIRCULAR** path from DC ($z = 1$) to $0.5f_s$ ($z = -1$).



4. Reconstruction (*-order hold, etc.)

- DSP-filtered signal exists abstractly as impulses. How to convert to continuous-time? Must somehow interpolate between points. Two common choices:



- What is the impact in continuous-time? How do we **convert** filters from z to s and back?
  - Need $s \leftrightarrow f(z)$ *close* to $z = e^{sT}$ with polynomial-ratio transfer functions in z *and* s.
  - One strategy (there are **many**) is to make sure integration (i.e., $1/s$) behavior is consistent.
  - To integrate, at each step, the cross-hatched area must be added to the previous sum.
    * The area at time $k+1$ under zero-order held rectangles (i.e., *forward rectangles*) is

$$y[k+1] = y[k] + Tx[k].$$

      So this discrete-time integrator has transfer function $T/(z-1)$. Because a continuous-time integrator is $1/s$, we substitute

$$s \longleftrightarrow \frac{z-1}{T} \qquad \text{and} \qquad z \longleftrightarrow Ts+1.$$

      · This "**forward rectangular**" transformation is *linear* (actually, it's *affine*).
      · This transformation maps $s \leq 0$ plane to the $z \leq 1$ plane (i.e., *not* just $|z| < 1$ circle).
      · Consequently, **all stable z-domain filters have stable s-domain approximations, but many s-domain filters do *NOT* have stable z-domain approximations.**
      · By increasing sampling rate (i.e., decreasing T), more stable s-domain poles get mapped inside the stable z-domain unit circle.

    * The area at time $k+1$ under first-order held *trapezoids* is

$$y[k+1] = y[k] + \frac{T}{2}\left(x[k+1] + x[k]\right),$$

      which has transfer function $(T/2)(z+1)/(z-1)$. Matching to $1/s$, we substitute

$$s \longleftrightarrow \frac{2}{T}\frac{z-1}{z+1} \qquad \text{and} \qquad z \longleftrightarrow \frac{2+sT}{2-sT}.$$

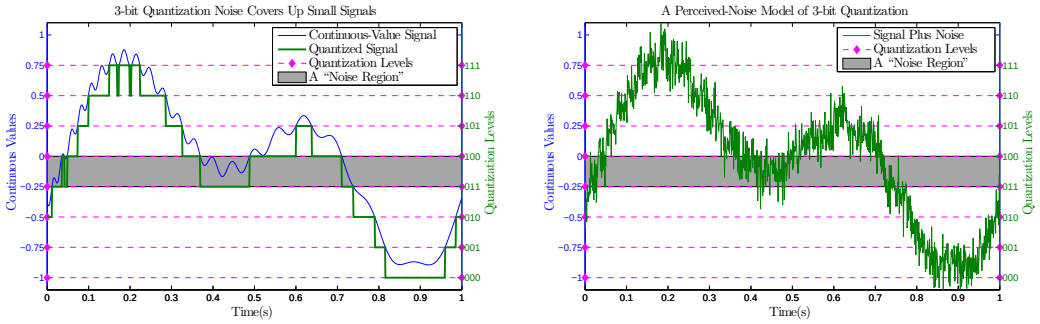
      · This **trapezoid rule** is a ratio of *two linear* maps; it is a "**bilinear transformation.**"
      · This bilinear transformation wraps the entire continuous-time frequency range (i.e., from $-\infty$ to ∞) to the discrete-time unit circle.
      · Consequently, **all stable s-domain poles are mapped to stable z-domain poles.**
      · In fact, the entire left-hand s-plane is mapped to the inside of the unit circle.
      · So stable s-domain filters will have stable z-domain approximations (and vice versa).
      · However, s-domain frequency response can be severely distorted at high frequencies from wrapping an infinite line around a single circle.
      · Increasing sampling frequency (i.e., decreasing T) increases size of low-distortion region.
      · It is also called "**Tustin's approximation.**"

5. Quantization

- Quantization both *distorts* a signal and *adds noise*. Because these two effects are not separable, the terms *quantization noise* and *quantization distortion* are often used interchangeably.
- Quantization *distortion* refers to the sharp edges added to signals. These sharp edges introduce new high frequency harmonics into the signal. Contrast the feel of a knob that can rotate continuously with a knob that "clicks" from one fixed position to another; the latter effect is noticeable, distinct, and possibly undesirable.
- Quantization also acts like a *noise source* that covers up small characteristics of a signal. The smallest change possible for a quantized signal is known as its *least-significant bit* (LSB), which is

$$\frac{V}{2^B}$$

where V is the full-scale range of a signal (e.g., 2 for a standard sinusoid that swings from -1 to 1) and B is the number of bits. Sub-LSB characteristics of this signal are lost after quantization. It is as if random noise was added to the signal — small changes are lost in uncertainty.



The *signal-to-noise ratio* (SNR) is the ratio of signal *power* to noise *power*, and it is usually expressed in deciBels (dB). We can estimate the noise power from quantization as 1 square LSB. So a good worst-case approximation of the SNR due to quantization is

$$10\log_{10}\frac{V^2}{\left(\frac{V}{2^B}\right)^2} = 20\log_{10}\frac{1}{\frac{1}{2^B}} = B \times 20\log_{10} 2 \approx B \times 6\,\text{dB}.$$

For each extra bit, the quantization noise gets cut in half, and so the SNR increases by 6 dB.

- Modern analog-to-digital converters can *shape* quantization noise through *dither* or Σ-Δ methods.
  - These methods greatly *oversample* so that the quantizer gets multiple "looks" at each sample. To oversample is to sample MUCH higher than is theoretically needed.
  - When a single sample falls between two quantization levels, the quantizer *randomly* chooses the upper or lower boundary. However, the probability of choosing either boundary increases as the distance to the boundary decreases (i.e., a value very close to one quantization level will often be given that level).
    * If the signal is *SLOWLY* changing, simply *averaging* the quantized output will recover it *exactly*. If oversampling is fast enough, this averaging may occur through the natural low-pass filters present in nature and no additional hardware will be needed.
    * Unfortunately, if the signal is *QUICKLY* changing, the sampler will not get many "looks" at each part of it, and the added noise will actually make it *HARDER* to recover. So it is important to *GREATLY* oversample.
    * So these methods actually *SHAPE* the quantization noise so that its SMALLER at low frequencies and higher at high frequencies.
  - These methods are so successful that 1-bit quantizers are usually all that is needed so long as the sampling is fast enough. These 1-bit quantizers are easy to implement (i.e., very cheap).

6. Complete the *Introduction to Digital Signal Processing* lab

   - In each task of the lab, you
     1. Generate a triangle wave or a sine wave at a specified frequency.
     2. *Acquire* it with the DSP's (12-bit) ADC.
     3. *Sample* the acquired signal at specified rate.
     4. Generate a continuous-time signal from the discrete-time one using a *zero-order hold* (ZOH).
     5. View the resulting signal in *dSPACE ControlDesk*.

     As you complete the lab,
     - Keep the Shannon–Nyquist theorem (e.g., aliasing effects) in mind.
     - Keep the different interpolation techniques (e.g., forward rectangular, Tustin) in mind.
     - Consider how a DSP can control a plant in response to feedback and reference signals.
     - Consider the negative effect of quantization on digital signal processing.

   - At several times, you will change the **sampling rate** of the **zero-order hold device**.
     - You **can ALWAYS** keep your fixed-step simulation time at 0.1 ms (i.e., 0.0001 s).

   - The lab manual instructs you to use an **external** function generator.
     - Use the BNC-to-RCA cable at your bench to connect the function generator to the ADC input on the *dSPACE* interface card.
     - **Alternatively,** you may use the *Triangle Wave* component in the *ECE 557 Simulink* library.
       1. Wire component to a *Saturation* block with 5 V limits.
       2. Wire saturation output to *gain* of 1/10.
       3. Wire gain to DAC output.
       4. Loop back DAC output port to ADC input port on interface card.
       5. The *frequency* can be adjusted within *dSPACE* using a numeric control.

       These steps cause the DSP to generate its own triangle wave. **MAKE SURE** your fixed-step simulation time is **NO GREATER than** 1 ms. A simulation time of 0.1 ms is sufficient.

       The standard ***Sine Wave*** source in *Simulink* can be used in a similar fashion.

   - You may use *Simulink's ZOH* block, but the sampling period cannot be modified within *dSPACE*.
     - Alternatively, use the *Sample and Hold* block from the *ECE 557* library.
     - Its sampling *frequency* can be adjusted within *dSPACE* using a numeric control.

   - **IF ANY** of the custom *ECE 557* library blocks are used, go to

     Simulation → Configuration Parameters → RTI variable description

     and turn on *Include mask and workspace parameters* and *save* and *build* your model. This change makes it easier to modify the *Triangle Wave* and *Sample and Hold* frequencies within *dSPACE*.

   - Tips:
     - Do **work** out of directory on **local** hard drive — use as MATLAB working directory.
     - In *Simulink*, the hotkey for building a model is Ctrl - B .
     - Start *dSPACE ControlDesk* before doing *Simulink* builds.
     - In *dSPACE ControlDesk*,
       1. Wire simState to 2-option radio button.
       2. Label one option Run and give value 2.
       3. Label other option Stop and give value 0.
       4. Set Capture Settings to automatically repeat runs and use *stop time* of 10 s.

       Then settings can be changed without restarting.
       * Use radio button to start simulation and pause if needed.
       * To modify numeric inputs, press Enter / ↵ to commit changes (i.e., don't *click* away).