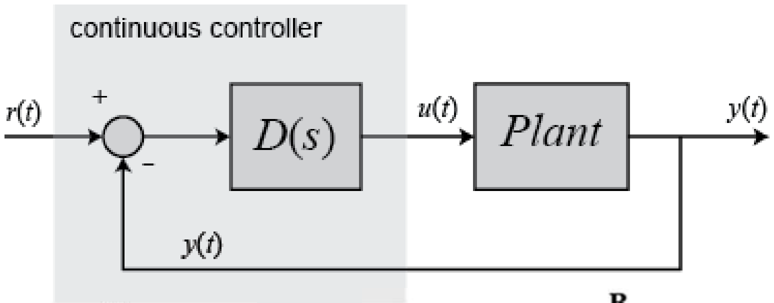
Discrete-Time & Digital Systems

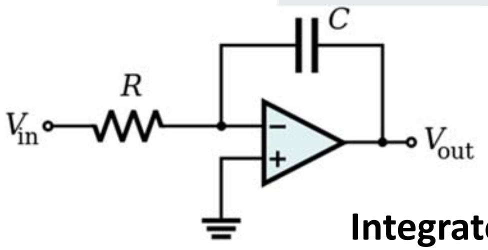
EPD 30.114 ADVANCED FEEDBACK & CONTROL

Continuous-Time Control

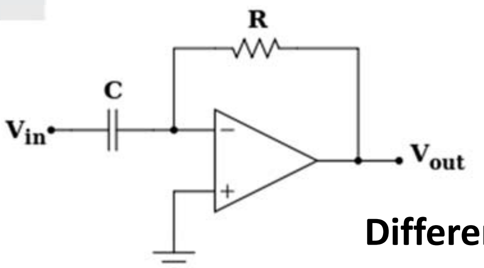
- Traditionally controllers were implemented using analog circuit components (Op Amps, Resistors, Capacitors, Inductors, etc)
  - This was great because analog devices are essentially operate on the principle of continuous time and can easily integrate and differentiate signals

- This allowed application of continuous-time (**Analog**) controllers to be integrated seamlessly with real world physical systems which are also continuous-time
  - Remember physical systems and continuous-time systems are described with Differential Equations!

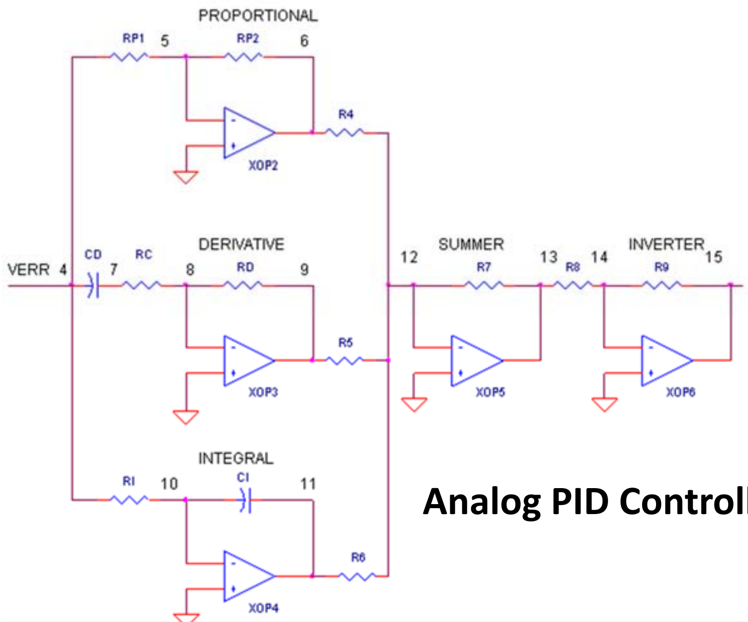




Integrator



Differentiator

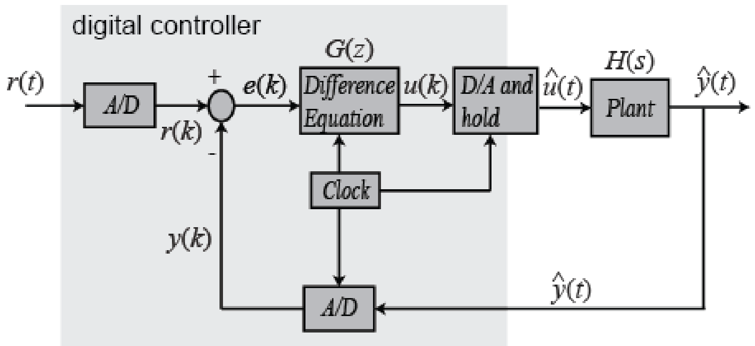


Analog PID Controller

Digital Control

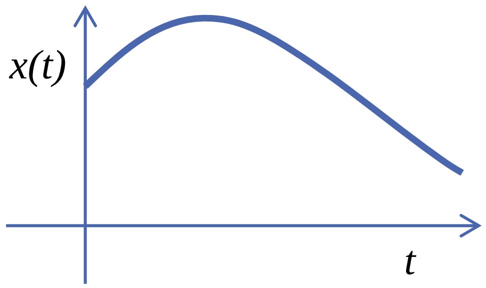
- There is a current trend towards digital control of dynamic systems due to availability of low-cost digital computers and advantages in working with digital signals (Noise attenuation, Error checking, low-power etc)
  - Complex, advanced and more intelligent controllers can be more easily implemented with digital components. They are also more flexible, allowing adjustment on the fly.

- However, there are 3 issues with Digital Systems:
  - Integration and differentiation are only approximated due to the finite sampling or clock. Hence **Difference Equations (Discrete-time)** are used to approximate differential equations (Continuous-Time)
  - Sampling and data hold also introduces an inherent **delay** into the system which has serious implication in closed-loop control
  - Lastly amplitude of digital signals suffer from **quantization error** due to the finite bits used to represent data (e.g. 10 bit D/A converter has 1024 values)

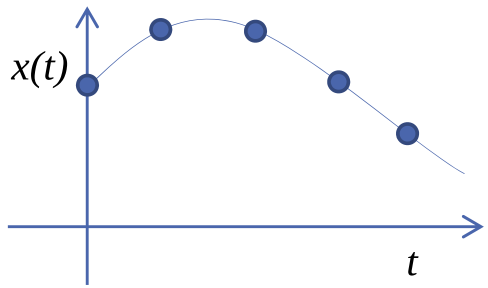


Recap of Signals

- Continuous-Time vs Discrete-Time Signal
  - Continuous-Time: Defined over a continuous range of time
  - Discrete-Time: Defined only at discrete instants of time (normally periodic)

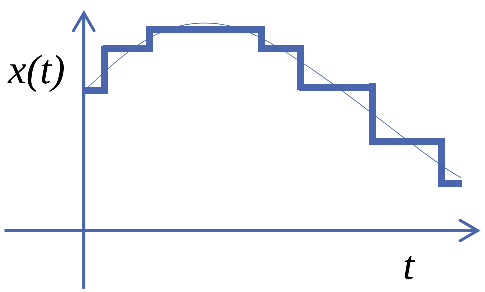


Continuous-Time Signal (ANALOG SIGNAL)

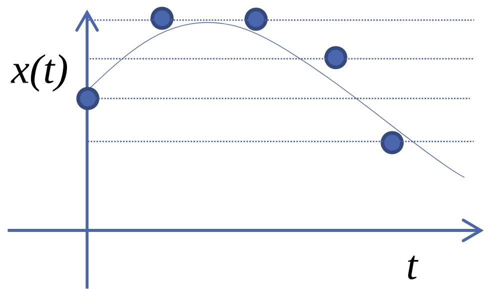


Discrete-Time Signal

- Amplitude Quantization
  - In a similar sense, the amplitude can either assume a continuous range of values or discrete



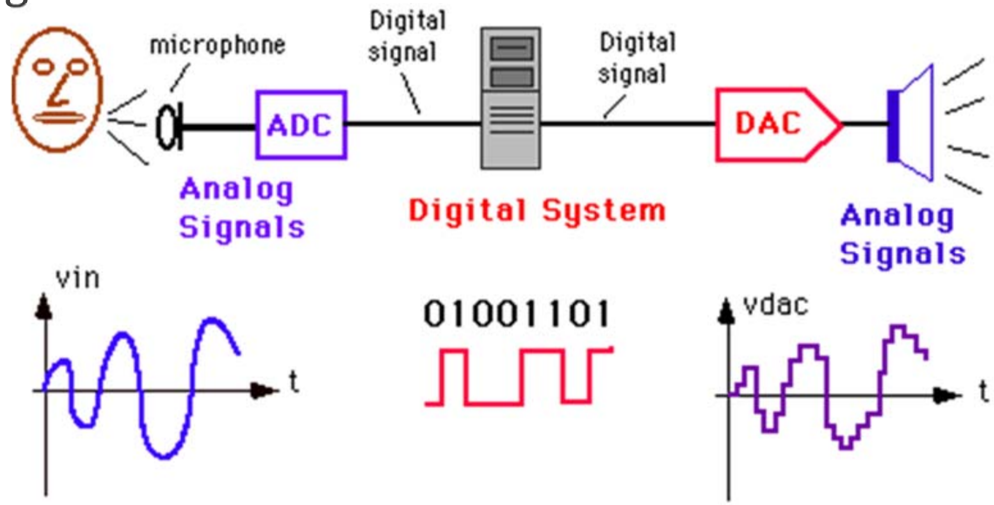
Continuous-Time Quantized Amplitude Signal



Discrete-Time Quantized Amplitude Signal (DIGITAL SIGNAL)

Interfacing Analog and Digital Signals

- An Analog-to-Digital conversion (ADC) is the process of converting a continuous physical quantity to a digital number representing the quantity's amplitude. It consists of, in chronological
  - Discretization/Sampling
  - Amplitude quantization
    - When the value of any sample falls between 2 adjacent 'permitted' output values, it will be assigned the permitted output nearest to the actual value of the signal
  - Encoding
    - The output state of each quantized sample is then described by a numerical code (normally a binary code).
- The Digital-to-Analog conversion (DAC) is the reverse function which converts a digital number to an continuous-time signal. It consists of:
  - Decoder
    - Transform a digital signal (Numerically coded data) into an analog signal



Discretization/Sampling Process

- The sampling of continuous-time signal replaces the original continuous-time signal by a sequence of values at discrete time points
- A signal whose independent variable *t* is discrete, is called a **discrete-time signal**
  - If the signal is further quantized and encoded, it is known as a **digital signal**
- The term 'Discretization' and 'Sampling' are used interchangeably though the former is frequently used in MIMO Systems (State-Space)
- There are several kinds of sampling operations:
  - **Periodic Sampling [Focus of this class]**
    - **Sampling instants are equally spaced**

$$t_k = kT \quad \left(k = 0, 1, 2, 3, \cdots\right)$$

  - Multiple-order Sampling
    - Pattern of t_k is repeated periodically
  - Multiple-rate Sampling
    - For control systems with multiple loops, there might be different sampling rate/period in each subsystem.
  - Random Sampling
    - Sampling interval is random.

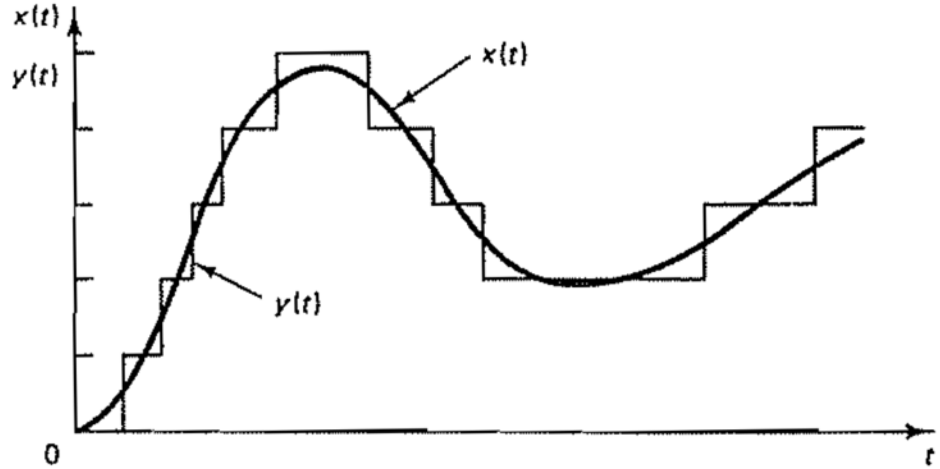
Quantization

- Computers operate on the binary number system (bits: on-off, 1 0)
- A system with *n* bits can represent 2^n discrete amplitude levels
- The Quantization level *Q*, is defined as the range between 2 adjacent decision points:

$$Q = \frac{FSR}{2^n - 1} = LSB$$

  - where FSR is the full-scale range.
  - In many cases, the least significant bit (LSB) is equal to the quantization level

- The quantization error associated with this approximation is *e(t)*
  - *x(t)*: Analog input
  - *y(t)*: Discretized output
  - *e(t)=x(t)-y(t)*

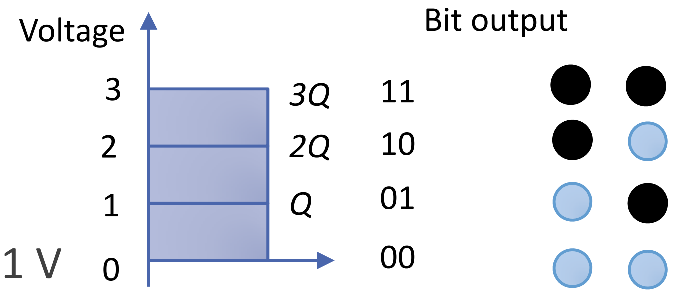


- Magnitude of this error is bounded:

$$0 \le |e(t)| \le \frac{1}{2} Q$$

Encoding

- With the quantized output, the last step is to relate it to a numerical code that can be processed easily in digital systems
- A very common method is using hexadecimals (base 16), due to its close relationship to the binary system (base 2). We humans use decimal system (base 10)
- Let's consider a 2 bit system used to represent 0 – 3 (voltage):
  - Total number of discrete amplitude levels the system can represent: 2^2
    - 00
    - 01
    - 10
    - 11
  - Quantization level: Q=(3-0)/(4-1)=1 V
  - Maximum quantization error is Q/2=0.5V



Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

- For a 4 bit system:
  - Discrete amplitude levels: 2^4 =16
  - Each level can be denoted by a single hexadecimal number (0-F)
- Today's ADCs are usually at least 10 bits [2^{10} = 1024 levels] (recall your Arduino!)
  - Good ones are 16 bit [2^{16} = 65536], professional at 24 bits [2^{24} = 16777216]

Numerical Approximation for Derivative

- Digital computers cannot integrate (they can only do multiplication and summation/subtraction), so to solve differential equations, the DE must be approximated

- One technique is to use numerical approximation which is derived from Euler's method
  - The definition of a derivative is: $\dot{x} = \lim_{\delta t \to 0} \frac{\delta x}{\delta t}$
  - For a sampled time function with sampling rate of *T:*

$$\dot{x}(k) \cong \frac{x(k+1) - x(k)}{T}$$

- This approximation can be used in place of all derivatives and implemented by a digital computer.
  - This version uses the **forward rectangular method** for integration approximation

Exercise

- Using Euler's forward method, find the difference equation to be used to approximate the following controller designed in s-domain:

$$D(s) = \frac{U(s)}{E(s)} = K\frac{s+a}{s+b}$$

The differential equation that corresponds to D(s) is:

$$(s+b)U(s) = K(s+a)E(s)$$

$$\dot{u} + bu = K\left(\dot{e} + ae\right)$$

Using the forward approximation:

$$\dot{x}(k) \cong \frac{x\left(k+1\right) - x\left(k\right)}{T}$$

$$\frac{u(k+1) - u(k)}{T} + bu(k) = K\left(\frac{e(k+1) - e(k)}{T} + ae(k)\right)$$

$$u(k+1) = u(k) + T\left[-bu(k) + K\left(\frac{e(k+1) - e(k)}{T} + ae(k)\right)\right]$$

$$u(k+1) = \left(1 - bT\right)u(k) + K\left(aT - 1\right)e(k) + Ke\left(k+1\right)$$

This equation shows how to compute the new value of control *u(k+1)* given the past value of control *u(k)*, and the new and past values of the error signal *e(k+1)* and *e(k)*

Forward and Backward Euler's Method

- The forward Euler's approximation was used earlier. But there is another approximation called the Backward Rectangular method:

$$\dot{x}(k) \cong \frac{x(k) - x(k-1)}{T}$$

- Do the previous exercise again but using the Backward approximation

$$\dot{u} + bu = K(\dot{e} + ae)$$

$$\frac{u(k) - u(k-1)}{T} + bu(k) = K\left(\frac{e(k) - e(k-1)}{T} + ae(k)\right)$$

$$u(k) - u(k-1) + bTu(k) = Ke(k) - Ke(k-1) + KaTe(k)$$

$$(1 + bT)u(k) = u(k-1) - Ke(k-1) + (1 + aT)Ke(k)$$

$$u(k) = \frac{1}{(1+bT)}u(k-1) - \frac{K}{(1+bT)}e(k-1) + \frac{(1+aT)}{(1+bT)}Ke(k)$$

$$u(k+1) = \frac{1}{(1+bT)}u(k) - \frac{K}{(1+bT)}e(k) + \frac{(1+aT)}{(1+bT)}Ke(k+1)$$

Introducing the Difference Equations

- Using various approximations, you can replacing derivatives (and will be shown later, integrals as well) so that you arrive with a set of equations that can be solved by a digital computer
- These equations are called Difference Equations and are solved repetitively with time steps of T
- In principle, the difference equation is evaluated initially at *k*=0 and then sequentially at *k*=1,2,3,…
- No requirement to store all values in memory and for first order difference equations, you only need variables from the current and past values
- A difference equation can be expressed generally as:

$$y(k) = f\left(u(0),\cdots,u(k);y(0),\cdots,y(k-1)\right)$$

- A **linear constant-coefficient difference equation (CCDE)** is:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \cdots - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + \cdots + b_m u(k-m)$$

- How to solve CCDEs?

Flashback! Solving ODEs!

- Let's say we have the following LTI ODE, with initial conditions. How to solve? $\ddot{y} = \dot{y} + y \qquad y(0) = 0, \;\; \dot{y}(0) = 1$
- Let's assume the solution to be of the form: $y = Ae^{at}$

$$y = Ae^{at}$$
$$\dot{y} = Aae^{at}$$
$$\ddot{y} = Aa^2e^{at}$$

$$\ddot{y} = \dot{y} + y$$
$$Aa^2e^{at} = Aae^{at} + Ae^{at}$$
$$a^2 = a + 1$$
$$a^2 - a - 1 = 0$$

This is also the c.e. why?

$$a = \frac{1 \pm \sqrt{1 - 4(-1)}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

$$a_1 = \frac{1+\sqrt{5}}{2} > 0$$
$$a_2 = \frac{1-\sqrt{5}}{2} < 0$$

Because the ODE is linear, if each of them is a solution, a linear combination of them is also a solution.

$$y = A_1e^{a_1t} + A_2e^{a_2t}$$

To solve for A_1 and A_2, we use the initial conditions!

$$y(0) = A_1 + A_2 = 0$$
$$\dot{y}(0) = A_1a_1 + A_2a_2 = 1$$

$$\Rightarrow A_1 = \frac{1}{\sqrt{5}}$$
$$\Rightarrow A_2 = -\frac{1}{\sqrt{5}}$$

$$y(t) = \frac{1}{\sqrt{5}}e^{\frac{1+\sqrt{5}}{2}t} - \frac{1}{\sqrt{5}}e^{\frac{1-\sqrt{5}}{2}t} \qquad t \geq 0$$

Ummm.. Better Approach!

- How about using our trusty Laplace Transform?

$$\ddot{y} = \dot{y} + y \qquad y(0) = 0, \;\; \dot{y}(0) = 1$$

$$s^2 Y(s) - sy(0) - \dot{y}(0) = sY(s) - y(0) + Y(s)$$

$$s_1 = \frac{1+\sqrt{5}}{2} > 0$$

$$\left(s^2 - s - 1\right) Y(s) = 1$$

$$s_2 = \frac{1-\sqrt{5}}{2} < 0$$

$$Y(s) = \frac{1}{s^2 - s - 1} = \frac{1}{(s - s_1)(s - s_2)} = \frac{1}{\left(s - \frac{1+\sqrt{5}}{2}\right)\left(s - \frac{1-\sqrt{5}}{2}\right)}$$

$$Y(s) = \frac{1}{\sqrt{5}} \frac{1}{(s - s_1)} - \frac{1}{\sqrt{5}} \frac{1}{(s - s_2)}$$

Taking Inverse Laplace Transform:

$$y(t) = \frac{1}{\sqrt{5}} e^{s_1 t} - \frac{1}{\sqrt{5}} e^{s_2 t} = \frac{1}{\sqrt{5}} e^{\frac{1+\sqrt{5}}{2} t} - \frac{1}{\sqrt{5}} e^{\frac{1-\sqrt{5}}{2} t} \qquad t \geq 0$$

unstable   stable

Solving CCDEs

- Ok good! Now consider an equivalent difference equation (*k* starts from 2): $$y(k) = y(k-1) + y(k-2) \qquad y(0) = 0,\ \ y(1) = 1$$
  - Just as the exponential was used for solving the LTI ODEs, we can use something similar as the solution of *y(k)*: $$y(k) = Az^k$$
  - Substituting: $$Az^k = Az^{k-1} + Az^{k-2}$$

$$1 = z^{-1} + z^{-2}$$

This is also the c.e. why? $$z^2 - z - 1 = 0$$

$$z = \frac{1 \pm \sqrt{5}}{2}$$

$$z_1 = \frac{1+\sqrt{5}}{2} > 0$$

$$z_2 = \frac{1-\sqrt{5}}{2} < 0$$

Because the CCDE is linear, if each of them is a solution, a linear combination of them is also a solution.

$$y(k) = A_1 z_1^{\,k} + A_2 z_2^{\,k}$$

To solve for A_1 and A_2, we use the initial conditions!

$$y(0) = A_1 + A_2 = 0 \qquad \Rightarrow A_1 = \frac{1}{\sqrt{5}}$$

$$y(1) = A_1 z_1 + A_2 z_2 = 1 \qquad \Rightarrow A_2 = -\frac{1}{\sqrt{5}}$$

$$y(k) = \frac{1}{\sqrt{5}} z_1^{\,k} - \frac{1}{\sqrt{5}} z_2^{\,k} \qquad k \ge 0$$

$$= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^k - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^k$$

Some Insights of the Linear CCDE

Leonardo Fibonacci

- Do you notice anything interesting about this equation?

$$y(k) = y(k-1) + y(k-2) \qquad y(0) = 0, \;\; y(1) = 1$$

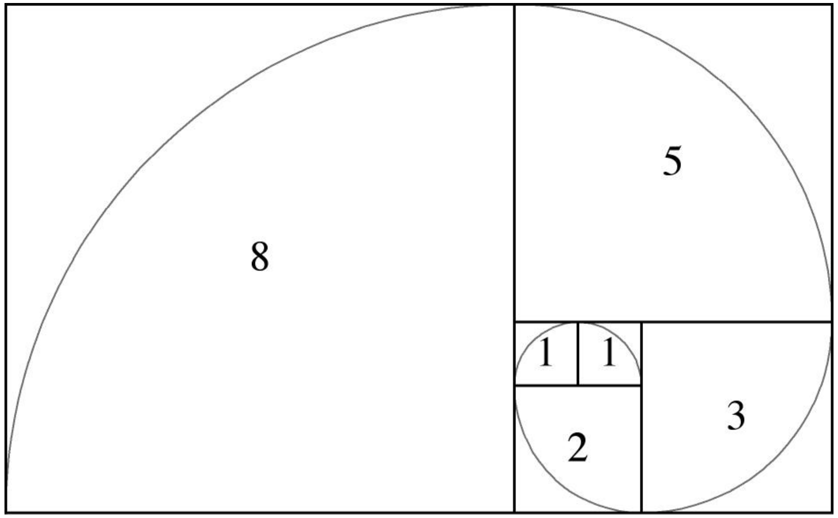
Yes! It is the Fibonacci Sequence!

As controller designers, we also wonder:
Is the sequence/system stable?

Recall that the characteristic equation of the difference equation is: $z^2 - z - 1 = 0$

Poles are at $z_1 = \dfrac{1+\sqrt{5}}{2} > 1 > 0$

$$z_2 = \frac{1-\sqrt{5}}{2} < 0$$



$$y(k) = \frac{1}{\sqrt{5}} z_1^{\,k} - \frac{1}{\sqrt{5}} z_2^{\,k} \qquad k \geq 0$$

$$= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^k - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^k$$

From the solution, you can see that if *z* is larger than 1, the expression z^k will grow without bounds.
For Stability of a CCDE, the poles must be within the unit circle about the origin

Great! But Is There Something Similar to LT?

- We managed to solve and analyze a 2nd order Difference Equation. But is there a method we could use to solve multi-order, input/output linear constant coefficient *Difference Equations* like how **Laplace Transform** could be used multi-order input/output LTI *Differential Equations* and handle initial conditions automatically?

- Yes! The z Transform is the mathematical tool used analyze and synthesize discrete-time control systems in the manner similar as to how the Laplace transform is used to analyzed and synthesize continuous-time control systems.

- In fact there's yet another intricate relationship between the ***s*-plane** and ***z*-plane** which we will explore in greater detail later!
  - As a sneak peak! https://www.youtube.com/watch?v=4PV6ikgBShw