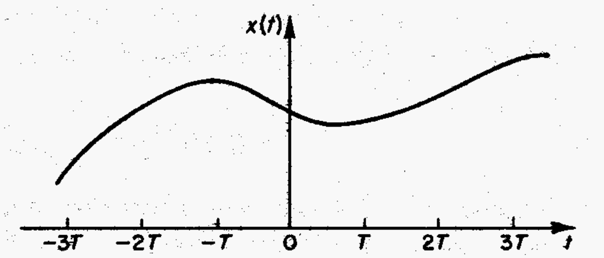
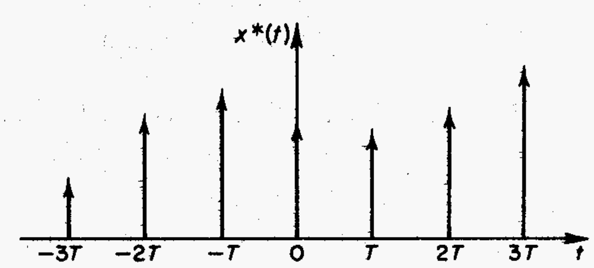
Introduction to Z-transforms

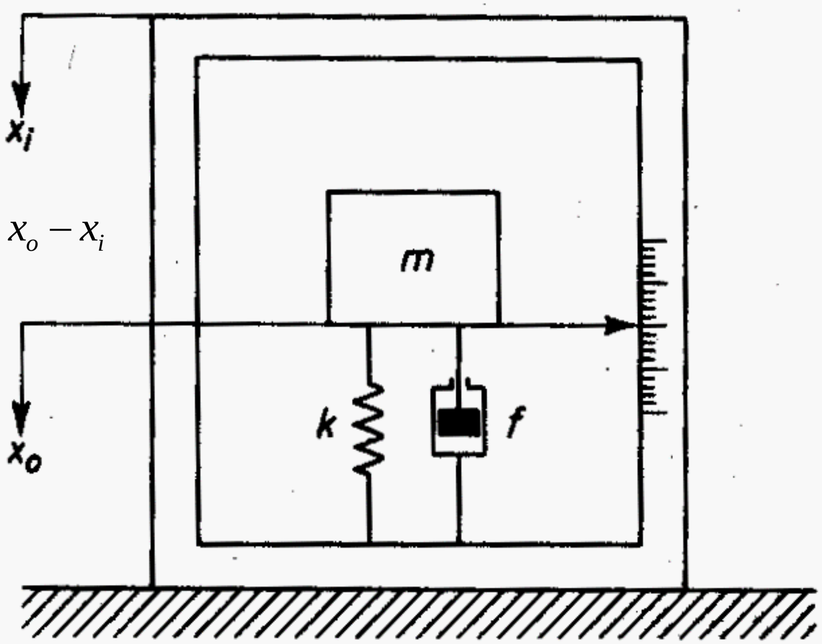


Performance Engineering of Real-Time and Embedded Systems



Continuous time functions vary smoothly and have values defined at all instants in time.

$$y = x_o - x_i$$

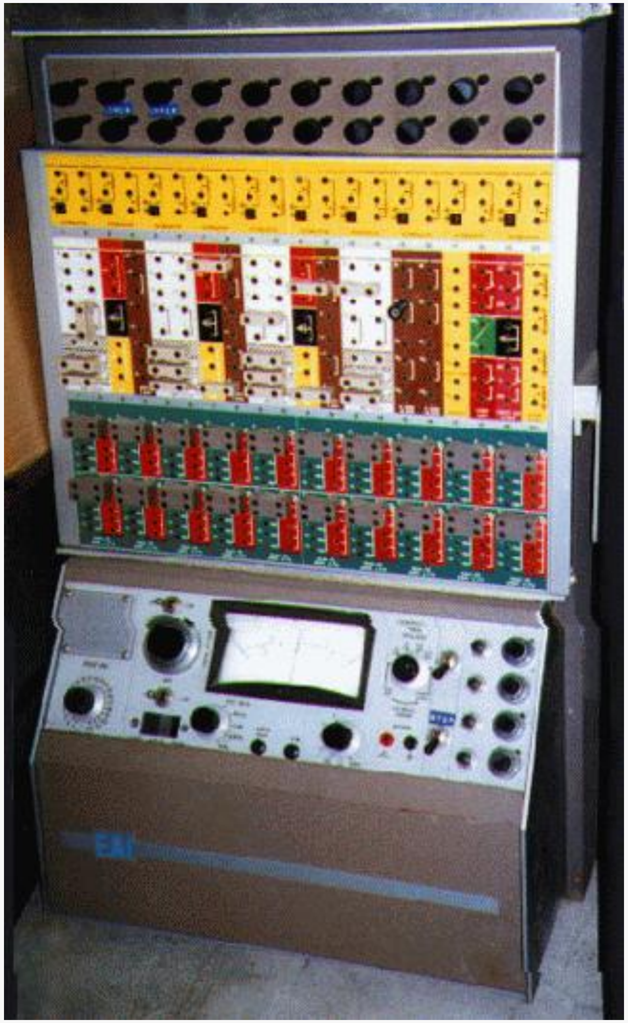


$$m\ddot{y} + f\dot{y} + ky = -m\ddot{x}_i$$

$$\frac{Y(s)}{X_i(s)} = \frac{s^2}{s^2 + \frac{f}{m}s + \frac{k}{m}}$$

Software Engineering

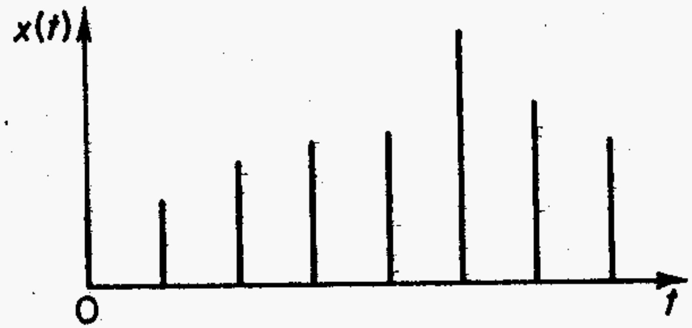
At one time, systems of continuous equations were solved on analog computers.

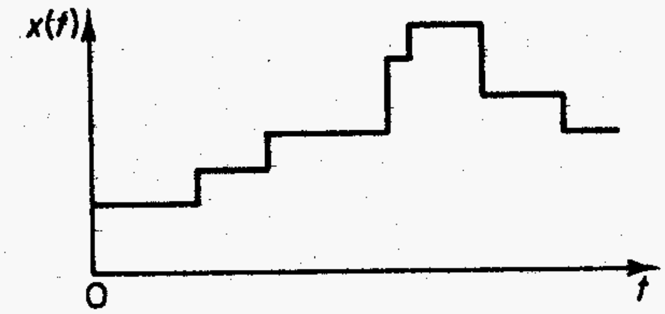


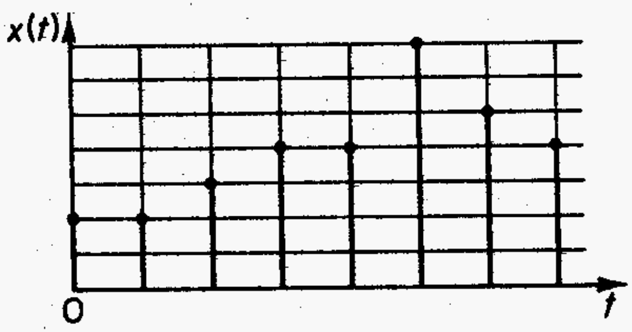
(Lexikon's History of Computing Encyclopedia on CD ROM)

Software Engineering

With digital computers came quantization in both time and value.







Software Engineering

We can handle discrete time systems in the time domain using difference equations.

An Integrator

$$y(k+1) = \sum_{i=0}^{k} u(i)$$

$$y(k+1) - y(k) = u(k)$$

$$y(k+1) = y(k) + u(k)$$

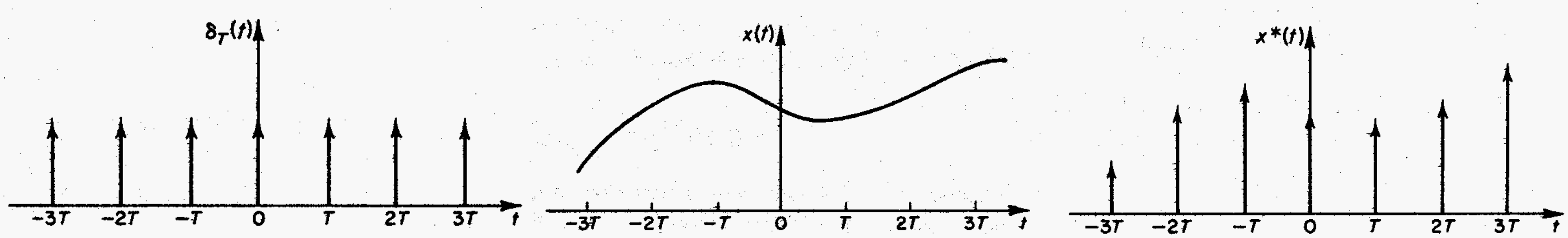
This is the equivalent of... staying in the time domain, i.e. differential equations, for continuous system analysis.

The time domain is again rather cumbersome so we do a transformation into the Z-domain.

$$\{u(k)\} = \{u(0), u(1), u(2), \ldots\}$$

$$U(z) = u(0)z^{0} + u(1)z^{-1} + u(2)z^{-2} + \cdots = \sum_{k=0}^{\infty} u(k) z^{-k}$$

The value of the signal with a Z-transform of *U(z)* at time *k* is the coefficient of *z*



There are closed form Z-transforms for many common signals.

$$U_{\text{impulse}}(z) = 1z^{0} + 0z^{-1} + 0z^{-2} + \cdots$$
$$= 1$$

$$U_{\text{step}}(z) = 1z^{0} + z^{-1} + z^{-2} + \cdots$$
$$= \sum_{k=0}^{\infty} z^{-k} = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$

$$U_{\text{ramp}}(z) = 0z^{0} + 1z^{-1} + 2z^{-2} + 3z^{-3} + \cdots$$
$$= \sum_{k=0}^{\infty} kz^{-k} = \frac{z}{(z-1)^{2}}$$

$$U_{\text{exp}}(z) = 1z^{0} + az^{-1} + a^{2}z^{-2} + a^{3}z^{-3} + \cdots$$
$$= \sum_{k=0}^{\infty} a^{k}z^{-k} = \frac{z}{z-a}$$

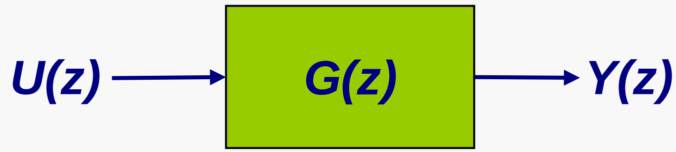
RIT Software Engineering

The control engineer’s toolbox has many analysis techniques.

- Inverse Z-transform
- Impulse response
- Analysis of zeros and poles, i.e. roots of the transfer function’s numerator and denominator

We will leave the detailed analysis to the control engineer and instead look at how to implement a transfer function and the empirical design of controllers.

It is easy to compute the output of a discrete transfer function given the input forcing values.



$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z^{n-1} + \cdots + b_m z^{n-m}}{z^n - a_1 z^{n-1} - \cdots - a_n}$$

$$Y(z)(z^n - a_1 z^{n-1} - \cdots - a_n) = U(z)(b_1 z^{n-1} + \cdots + b_m z^{n-m})$$

$$z^n Y(z) = a_1 z^{n-1} Y(z) + \cdots + a_n Y(z) + b_1 z^{n-1} U(z) + \cdots + b_m z^{n-m} U(z)$$

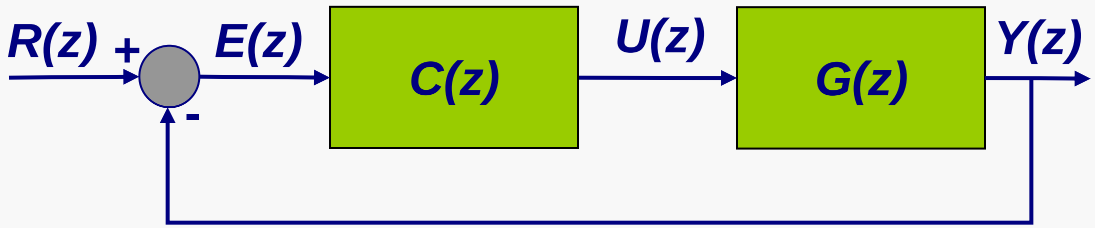
$$zY(z) = a_1 Y(z) + \cdots + a_n z^{1-n} Y(z) + b_1 U(z) + \cdots + b_m z^{1-m} U(z)$$

zY(z) is the output at the next sampling instant. In the time domain, this is $y(k+1)$.

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

Why should the highest z exponent in the numerator never be greater than the highest z exponent in the denominator? Can they be the same power?

A standard controller is the Proportional-Integral-Derivative (PID) Controller



$$C(z)=\frac{U(z)}{E(z)}=\frac{(k_p+k_i+k_d)z^2-(k_p+2k_d)z+k_d}{(z-1)z}$$

http://www.wescottdesign.com/articles/zTransform/z-transforms.html

- k_p drives plant proportional to the error
- k_i drives plant proportional to the integral of the error
- k_d drives plant proportional to the derivative of the error

There are heuristic guidelines for adjusting the constants in a PID controller.

As you increase the contribution of each element in the controller by increasing its constant you see these effects.

Response	Rise Time	Overshoot	Settling Time	Steady State Error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Large Decrease
Kd	Minor Decrease	Minor Decrease	Minor Decrease	No Change

http://en.wikipedia.org/wiki/PID_controller#Manual_tuning

The effects are not completely independent however.