

**There are Multiple Methods of Discretization, Including:**

- Backward difference
  - Approximates derivative
- Forward Difference
  - Approximates integral
- Combined Forward/Backward Difference
  - Better derivative estimate
- Bilinear transform
  - End point trapezoidal integration
- Impulse invariance
  - Sampled version of impulse response
- Transition Matrix Method
  - Solve first order matrix differential equation
- Bootstrap Method (overkill)
  - Use  $x_k$  and  $x_{k+1}$  to compute  $x'_k$
  - Use forward difference to compute  $x_{k+1}$
  - Use backward difference to compute  $x_{k+1}$
  - Use bilinear transform to recompute  $x_{k+1}$

© 2000 Kip Haggerty, ARR

1

**Multiple Methods of Discretization Provide Slightly Different Difference Equations**

- Backward difference  $\dot{y}_k = \frac{y_k - y_{k-1}}{T}$   $s = \frac{1-z^{-1}}{T}$   $z = \frac{1}{1-sT}$
- Forward Difference  $\dot{y}_k = \frac{y_{k+1} - y_k}{T}$   $s = \frac{z-1}{T}$   $z = 1+sT$
- Combined Forward/Backward  $\dot{y}_k = \frac{y_{k+1} - y_{k-1}}{2T}$   $s = \frac{z-z^{-1}}{2T}$
- Bilinear transform  $\frac{\dot{y}_{k+1} + \dot{y}_k}{2} = \frac{y_{k+1} - y_k}{T}$   $s = \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}}$   $z = \frac{1+\frac{T}{2}s}{1-\frac{T}{2}s}$
- Impulse invariance  $z = e^{sT}$
- Transition Matrix Method  $\dot{y} = A y(t) + B x(t)$   
 $y_{k+1} = e^{AT} y_k + \int_0^T e^{A(T-t)} B x(kT+t) dt = y_k + k$

© 2000 Kip Haggerty, ARR

2

**Example: A simple Integrator**

- Backward difference  $y_k = y_{k-1} + T x_k$
- Forward Difference  $y_k = y_{k-1} + T x_{k-1}$
- Bilinear transform  $y_k = y_{k-1} + \frac{T}{2} (x_k + x_{k-1})$
- This can lead to:
  - Time wasting arguments
  - Confusing requirements
  - Questions about which transformation to use for each compensator
  - Incorporation of transformation into delivered code
  - Software design and coding errors
  - Wasted time in integration

© 2000 Kip Haggerty, ARR

3

**Bilinear Transform Produces All First Order Compensators and Filters from the Same General Model**

- Continuous Time Transfer Function  $H(s) = k \frac{z^p + u_z}{p + u_p}$  where:  $u_z, u_p = [0 \text{ or } 1]$
- Bilinear Transformation  $s = \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}}$  where: T = updatetime
- Discrete Time Transfer Function  $H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}}$  where:  $b_0 = k \frac{2 - u_z + u_p}{2 - u_z - u_p} \frac{T_s}{T_s}$   
 $b_1 = -k \frac{2 - u_z - u_p}{2 - u_z + u_p} \frac{T_s}{T_s}$
- Difference Equation  $y_n = a_1 y_{n-1} + b_0 x_n + b_1 x_{n-1}$

© 2000 Kip Haggerty, ARR

4

**Same Difference Equation Provides 6 Different First Order Compensators Depending on Model Parameters**

Type:	k	$\tau_z$	$u_z$	$\tau_p$	$u_p$
Lead Compensation	$\neq 0$	$> \tau_p$	1	$> 0$	1
Lag Compensation	$\neq 0$	$> 0$	1	$> \tau_z$	1
Low Pass Filter (LPF)	$\neq 0$	$= 0$	1	$> 0$	1
Differential LPF	$\neq 0$	$= 1$	0	$> 0$	1
Integrator	$\neq 0$	$= 0$	1	1	0
Proportional plus Integral (PI)	$= k_I \neq 0$	$k_p/k_I \neq 0$	1	1	0

© 2000 Kip Haggerty, ARR

5

**Clamping and Preset is Accomplished by Modifying the Stored State Regardless of Type**

- Accomplished by Changing the Stored Previous State Before Next Iteration  

$$y_n = a_1 y_{n-1} + b_0 x_n + b_1 x_{n-1}$$
- Clamping:
  - Clamp  $y_n$  before output and save as previous value
  - Output is clamped
  - On next iteration, previous state is clamped (no build-up)
  - For PI, Equivalent to separating out Integrator and clamping its state
- Preset:
  - Compute initialization value
  - Store as previous state before first or next execution

© 2000 Kip Haggerty, ARR

6