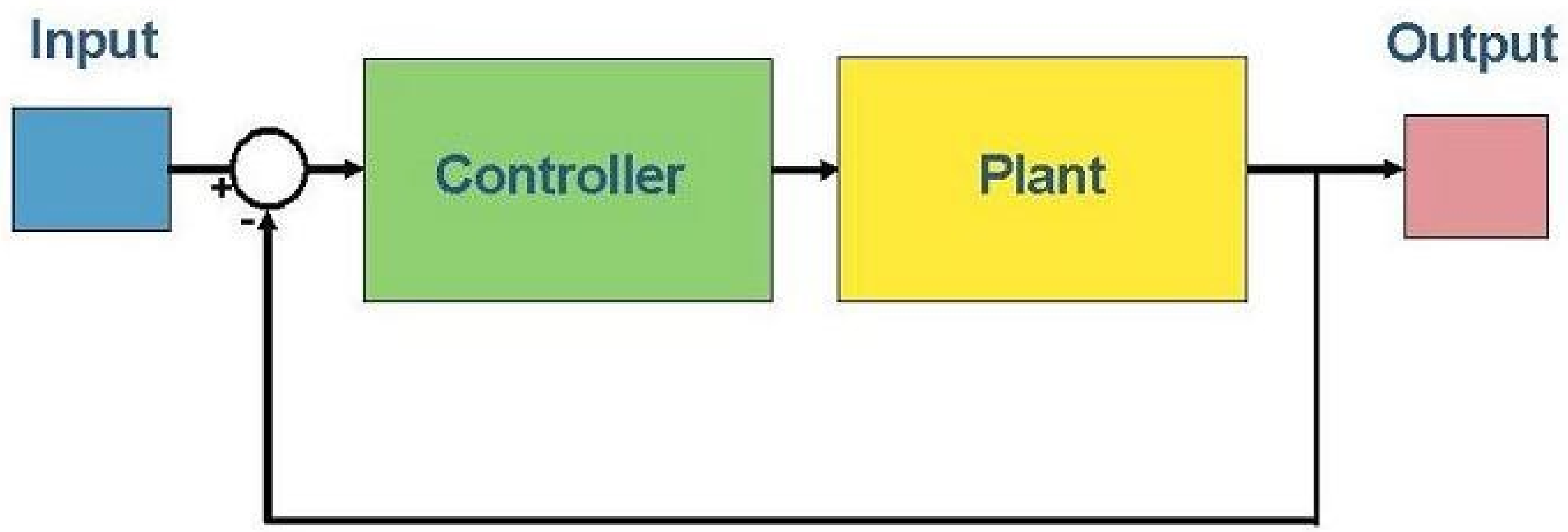
ESP Servo Motor Control Overview

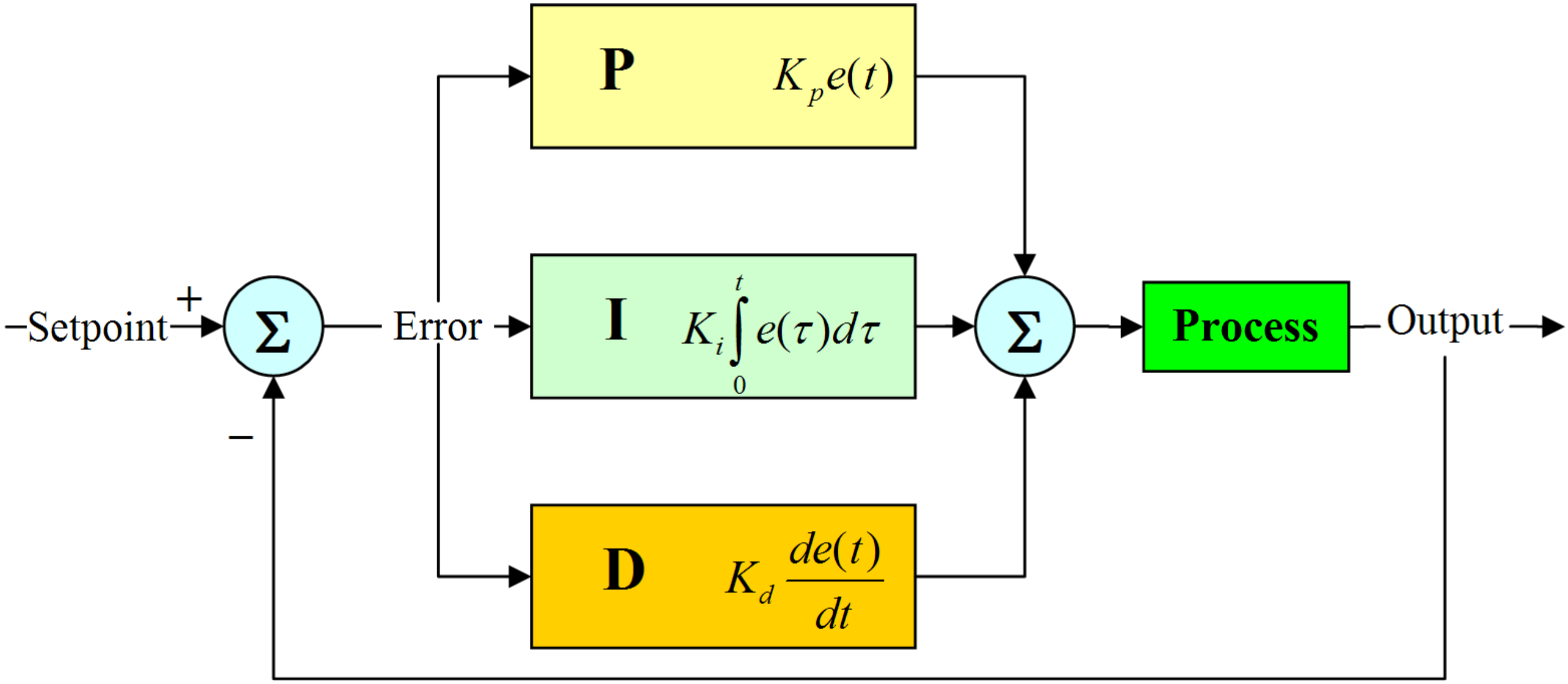
3/10/10 Brent Roman brent@mbari.org

M B A R I

Generic
Control Block Diagram

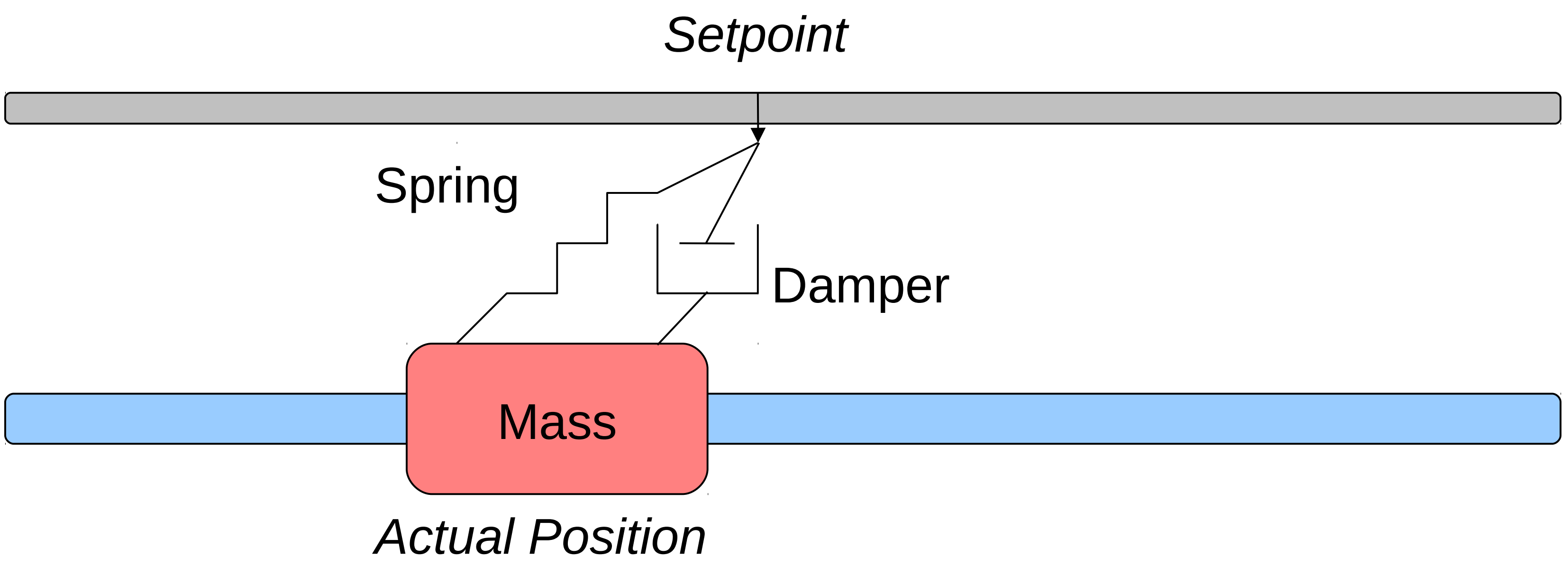


Classic
Proportional, Integral, Derivative (PID)
Control Block Diagram



3 simple, independent, parallel linear control laws

Proportional, Derivative
Mechanical Schematic Diagram

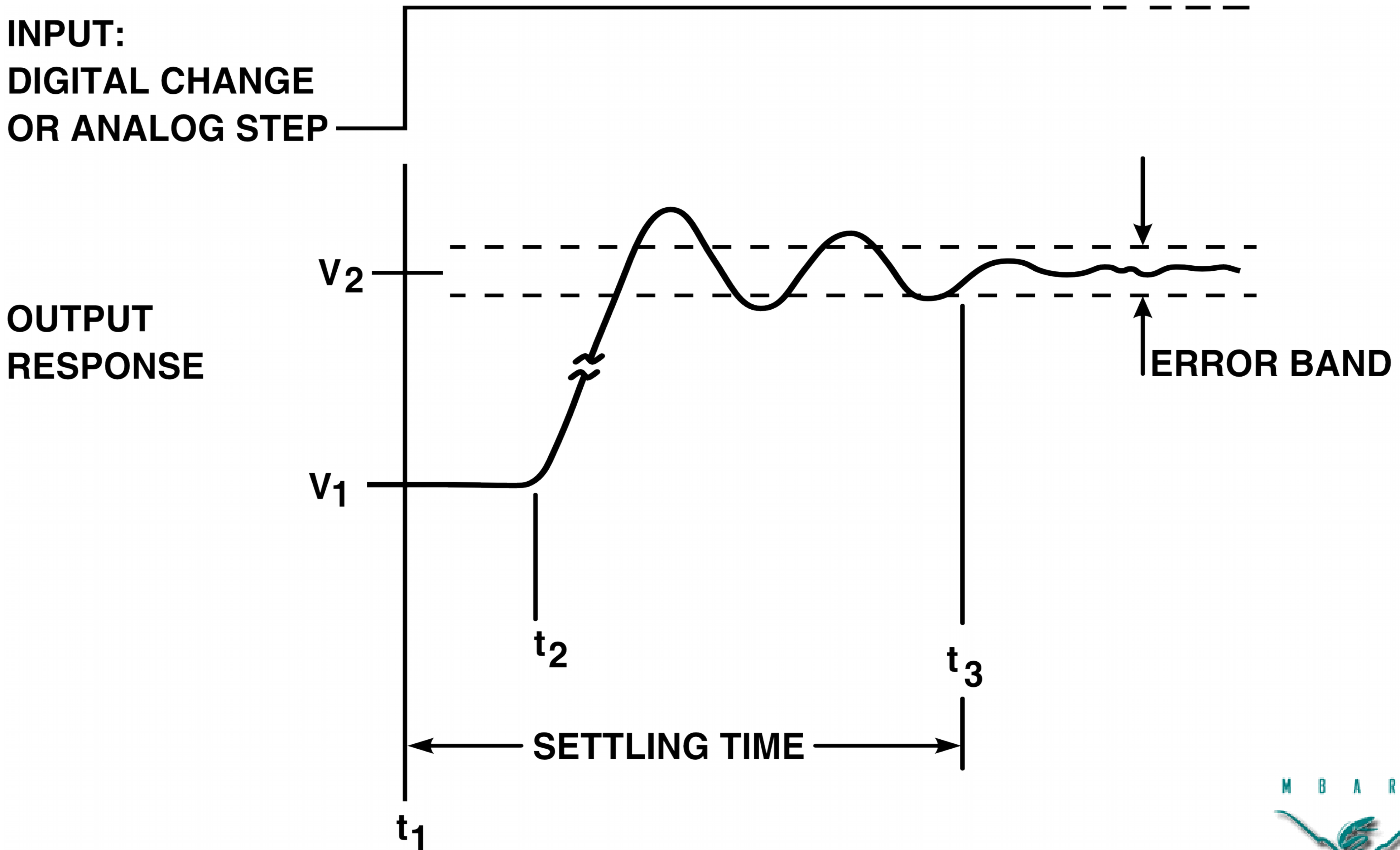


P is the spring constant, D is the degree of damping stiffness

Taut spring → Large control forces & fast response/ringing
Loose spring → Small control forces & slow response/ringing
Damper reduces ringing

Step Response

Trace of Mass motion after setpoint suddenly changed

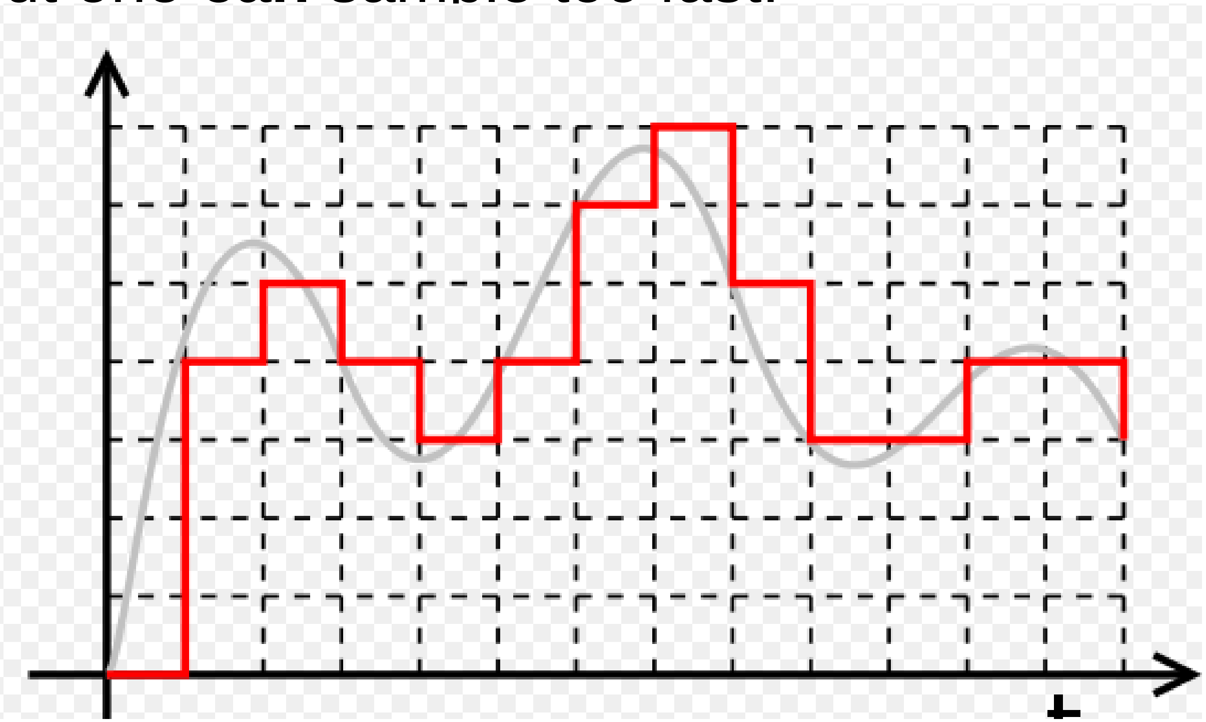


Tuning P and D gains
for a desired step response

- Analogous to “tuning” a car/truck/motorcycle suspension
- Higher Proportional Gain (P) stiffens spring
  - Faster response with higher frequency ringing
  - More peak power required
- Higher Derivative Gain (D) increases active damping
  - Can be very effective at reducing ringing
  - Even more peak power required!
  - Sudden changes in setpoint cause faster corrections
- All noise is amplified as well
  - from sensors, mechanism or actuator
  - System “unstable” when noise fed-back with gain >1

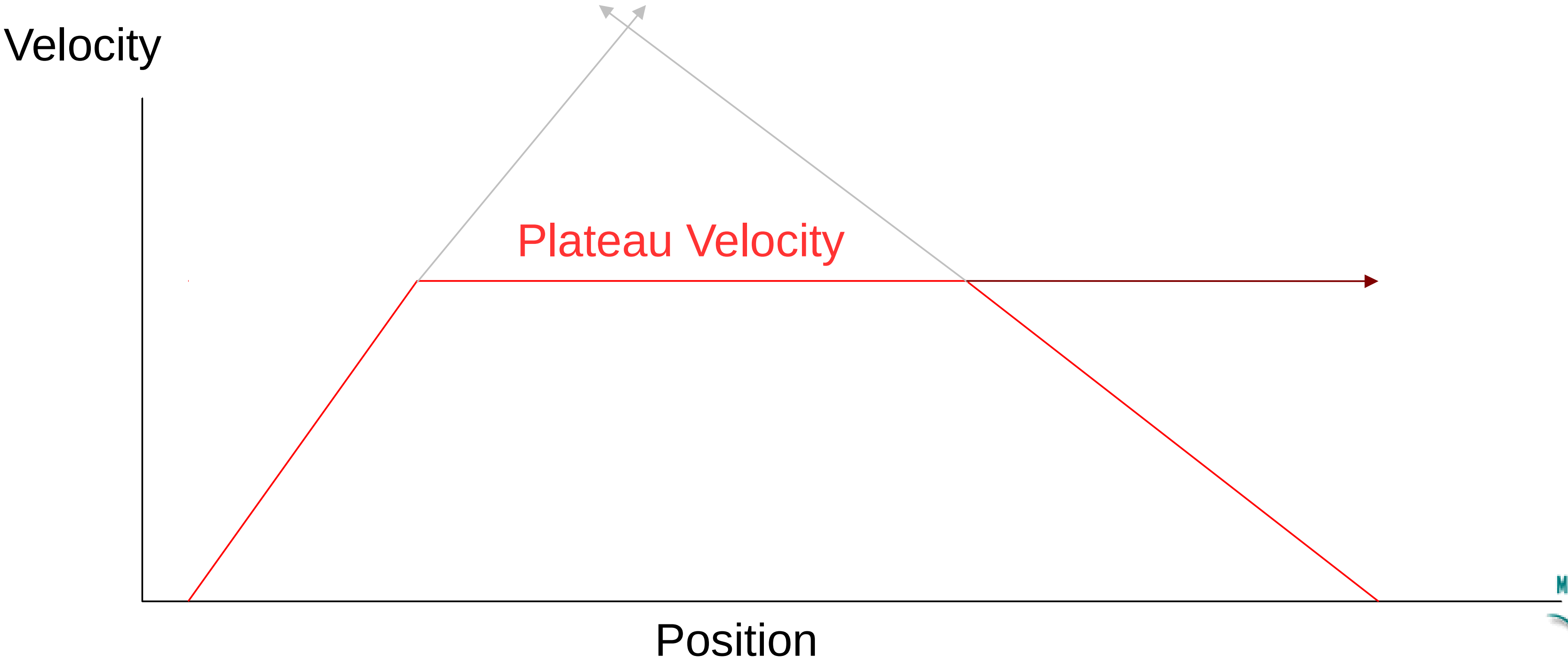
Discretization Errors

- Discretization occurs everywhere in digital controllers
  - Input, Output, and even Time itself
- Must sample at rate least twice ringing frequency
  - 5x is usually better in practice
  - But one **can** sample too fast!



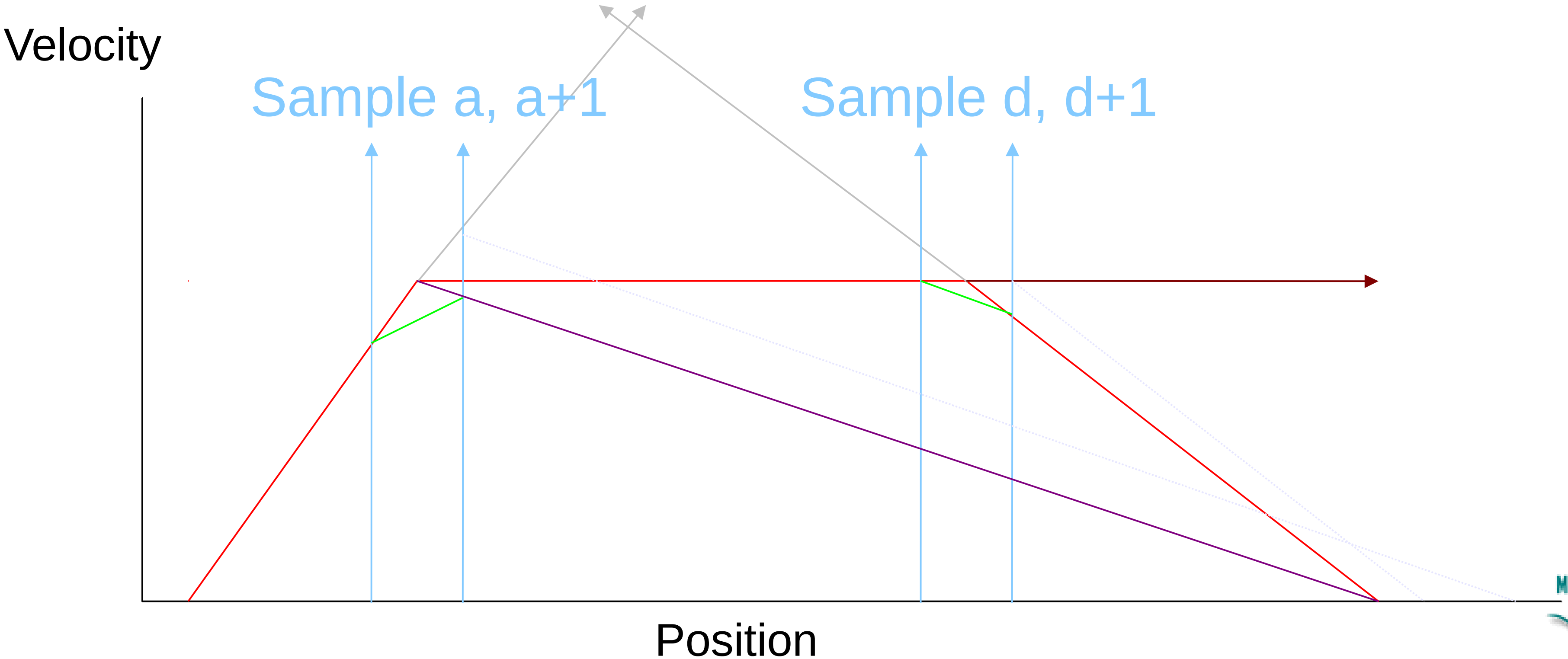
Trajectory Generator Basics

- Moves setpoint (gradually) from one goal to the next
- Why bother?
  - Because we don't have infinite power (yet!)
  - Allows tight control of velocity profile



Trajectory Generator Complications

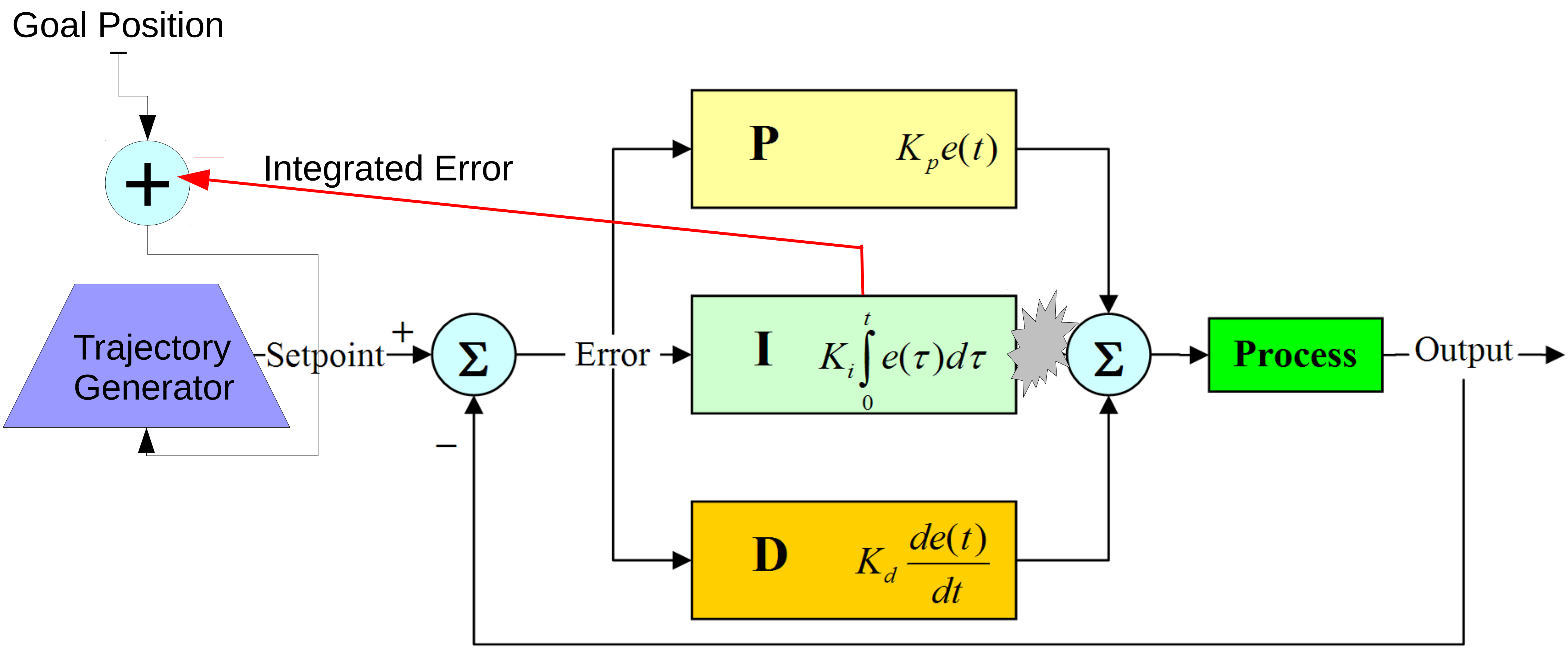
- Discretization and differing acceleration / deceleration require special care



Why not use the I-term to Eliminate Steady-state Error?

- Every PD controller must see a non-zero error to generate a corrective force.
- Turning the PD gains up to minimize this may not be practical.
- Feed back the err integral to gradually drive steady-state error to zero!
  - That's what the I-term of the PID does, right?
  - Yes. At the cost of reduced bandwidth, increased overshoot, hunting, etc.
  - Increasing the I gain enough to make it useful will often add noticeable ringing to the servo's step response (or even destabilize it)
  - Consider what happens after pushing against a stop for an extended period, once that stop is removed, or whenever the output saturates.

ESP's Modified PID
Control Block Diagram



Integrator output redirected offset goal input to trajectory generator

M B A R I

Letting the Trajectory Generator Eliminate Steady-State Error

- Who says the trajectory must stop at the stated goal?
- Keep moving setpoint until the actual goal is reached
  - As long as it doesn't deviate from it too much
- Does not cause ringing or destabilize the PD servo loop
  - Unless you go crazy with the I-gain
- Error is expressed in terms of a position offset rather than a hidden state variable.
- Add a deadband to prevent hunting
  - This helps the transitional integrator approach too

Stiction

- PID can be optimal only for linear systems
  - No real system is completely linear!
- Small actuators usually have significant initial stickiness
- Causes jerkiness at the start and end of seeks
  - Effectively limits minimum smooth running speed
- High D gain and sample rates may help mitigate
- Applying an inverse nonlinear compensation that adds a constant force whenever command velocity is non-zero may quickly “break stiction” if its degree is repeatable
- There are many more techniques...no magic bullet.

Back EMF

- Motors are generators are motors, yada, yada...
- Control laws are formulated in terms of a force
  - PWM or voltage is not directly related to output force
  - Motor Current is!
    - High end servos regulate current at high bandwidth
    - This is the “right” way to cope with back-EMF
    - But, it takes a DSP or dedicated analog control loop
  - ESP compensates somewhat with a fed-forward gain on observed velocity called “friction” compensation.
  - We could do this better.

Adding an outer Pressure Servo

- Pressure is a proxy for any analog input to be regulated while moving from one goal position to another
- Never move backward, even if pressure is increasing
  - Is this constraint application specific to the ESP?
- Stops if pressure exceeds configured “safe” limit
- Simple Proportional controller (only a P-gain)
  - Acceleration on trajectory is set proportional to pressure error, subject to trajectory's constraints
  - No attempt to drive pressure error to exactly zero

ESP Dwarf DC Motor Servos

- Two identical servo channels
- 64hz sampling timebase (sample rate typically 32hz)
- Each Channel's Inputs:
  - Quadrature incremental encoder
    - (A and B 90 degrees out of phase)
  - Home flag (typically a hall effect sensor)
  - Optional threshold sensor
  - Forward and Reverse limit switches
  - One General Purpose digital input bit (for gripper)
- Each Channel Outputs:
  - PWM -100% to 100% (15 kHz with 1% resolution)
  - One General Purpose digital output bit

Units: No Floating Point

- MSP430 would not be able to compute floats fast enough
- Avoids whole issue of round off errors
- P and D gains expressed as 16-bit integers/4096
- Positions are 32-bit encoder counts relative to “home” flag
- Time expressed in “tics”
  - Each tic corresponds to one controller sample update
  - Typically 32hz or 64hz (but could be any submultiple)
- Velocity expressed in 16-bit encoder counts per tic
  - Ensure nothing ever moves faster than 32000/counts/tic!!
- Acceleration expressed as counts/tic/tic
- Electrical Current expressed in milliamps
- Pressure expressed in ADC counts (application must convert)

Basic Configuration

- :samplePeriod = number of 64hz timebase tics per sample tic
  - Default value = 2 (Typically 1 or 2)
- :encoder, :threshold, :home sensor power / polarity
  - Default value = :off (may be :positive or :negative)
- :homeDirection = :forward or :reverse
  - Default value = :reverse
  - :reverse moves negative if home flag inactive
- :brake = short motor terminals on servo error (:false or :true)
  - Default value = true
- :debug = output servo state at sample rate while seeking goal
  - Default value = false

Control Gains and Factors

- PID :gain struct with members P, I, and D
  - Default values for each are 0
  - Servo will not operate until at least one is non-zero
  - Effective value of P and D is divided by 4096
  - I is effectively divided by 16384
- :friction compensation gain
  - cmdVel * friction / 4096 added to PWM output
  - cmdVel = Commanded velocity
- :stiction compensation factor
  - If negative cmdVel, subtract stiction/2 from PWM
  - If positive cmdVel, add stiction/2 to PWM

Trajectory Generator (1 of 2)

- :acceleration & :deceleration in counts/tic/tic
  - Default values for each are 0, normally positive
  - Specify negative acceleration to disable “softstart”
  - Zero :deceleration implies deceleration=abs(acceleration)
- :maxSpeed = plateau velocity in counts/tic
  - Temporarily reduced when PWM limits reached to prevent trajectory errors due to low battery voltage
- :minSpeed = slowest acceptable progress rate (counts/sec)
  - Speed error if maxSpeed reduced below minSpeed
- :maxSettling = max tics to allow to servo to settle at goal
  - Default 0, typically 2 – 3 seconds worth of tics
  - Just ensures that position error not returned too early

Trajectory Generator (2 of 2)

- :stopWindow detemines how nearly goal should be reached
  - Specified in encoder counts (16 bit limit max)
  - Temporarily increased each time goal is passed
  - Special Value false indicates no (more) reseeks allowed
  - Defaults to Special Value :deceleration = deceleration rate
  - Also accepts value :acceleration

- :hunt determines whether to adjust setpoint after goal reached
  - Defaults to false, set true to “fight” to hold exact position at goal
  - Setpoint is *never* adjusted if position within stopWindow

- :thresholdOffset determines how far from threshold to stop when reached
  - Defaults to 0 encoder counts
  - When threshold reached before goal, goal = position + thresholdOffset
  - Used to position top of puck stack with respect to ESP's top plate

Core Limits

- :maxPWM & :minPWM
  - Max must be >= min, but each may be negative or positive
  - Constrains servo output, but does not constrain “force” command
  - Effective maxSpeed is reduced when servo reaches these PWM limits

- :maxPositionErr determines absolute maximum tolerable servo error in different contexts:
  - SeekErr if stopWindow grows too large due to repeatedly missing goal
  - TrajectoryErr if position becomes too far from setpoint while transiting
  - PositionErr if position moves too far from goal after arrival

- :maxCurrent determines maximum allowable motor current
  - In milliamps
  - Should never be set > 2000mA

Pressure Limits

- :maxInPress, :maxOutPress, :minInPress, :minOutPress
  - 0 to 4095 ADC counts
  - Maximum/Minimum tolerated Intake and Outlet pressures
  - Constraint disabled if corresponding max == min
  - All default to 0
- :maxDeltaPress & :minDeltaPress -- (-4095 to 4095)ADC counts
  - Maximum/Minimum tolerated pressure difference
  - Constraint disabled if set to special value: false
  - All default to false (there is no corresponding value true)
- Generic “Pressure Error” results if any of the above are violated
  - One must check status to determine the exact problem

Pressure Servo Configuration

- :inputDeltaPress determines if pressure delta is sensed or derived
  - True to input the difference from ADC 7
  - False to derive it as (intake – outlet) pressure
  - Defaults to false
- :pressBias is subtracted from delta pressure before use
  - In servo or limit check
  - Defaults to 0
- :pressGain is the proportional gain of the pressure servo
  - Scaled like P and D, pressGain is *4096
  - Reduces acceleration from that normally determined by the trajectory generator.
  - Never causes command velocity to fall below minSpeed

Servo Status

- :enabled = true if servo control is active
- :pastRLS, :pastFLS, :pastThreshold, :home
  - True if corresponding switch is closed
- :position = 32-bit signed offset from home position
- :velocity = 16-bit signed in encoder counts/tics
- :current = signed milliamps
  - Always agrees with sign of PWM status below
- :PWM = signed percent PWM duty cycle
- :err = 16-bit signed (setpoint – position)
- :voltage = raw motor voltage (in volts)
  - This is the *only* floating point value

Servo Pressure Status

- Recall that pressure may be a proxy for any arbitrary volage input
- :inPress = intake pressure in raw ADC counts (0-4095)
- :outPress = outlet pressure in ADC counts
- :deltaPress = delta pressure in ADC counts
  - This is always ADC channel 7
  - It is *not* affected by the :inputDeltaPress configuration flag

M B A R I