Sensors and Sensing
Motors, Encoders and Control

Todor Stoyanov

Centre for Applied Autonomous Sensor Systems
Örebro University
Sweden

Outline

1. Motors
2. Encoders
3. Motor Control

Outline

1. Motors
2. Encoders
3. Motor Control

Motor Basics

- A motor is a device which transforms input energy into kinetic energy.
- Input energy could be chemical (e.g. gasoline engine), but we are interested in electro-magnetic motors.
- First electric motors were invented in the 19th century.
- Hungarian physicist Ányos Jedlik demonstrated the first electric motor.

Motor Basics

- A motor is a device which transforms input energy into kinetic energy.
- Input energy could be chemical (e.g. gasoline engine), but we are interested in electro-magnetic motors.
- First electric motors were invented in the 19th century.
- Hungarian physicist Ányos Jedlik demonstrated the first electric motor.

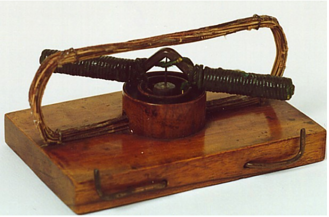
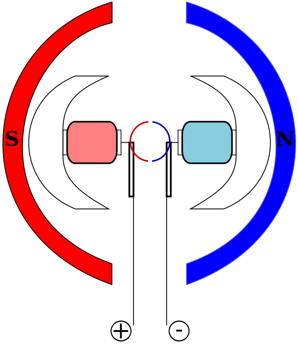


Figure: Image source: wikipedia

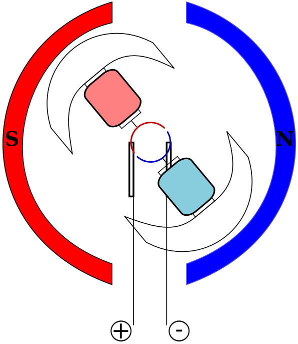
Brush Motors

- Brush motors were the predominant type of motor until the 1960s.
- The design includes three key components: the stator, the rotor and the commutator.
- Stator is typically implemented as a permanent magnet.
- Rotors are coils which act as magnets when electricity flows through them.
- The commutator is a mechanical device used for switching the direction of electric current.



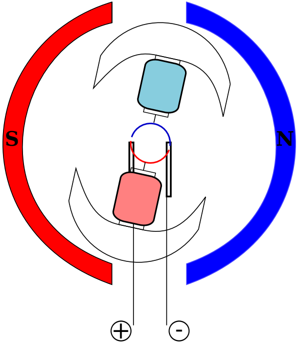
Brush Motors

- Brush motors were the predominant type of motor until the 1960s.
- The design includes three key components: the stator, the rotor and the commutator.
- Stator is typically implemented as a permanent magnet.
- Rotors are coils which act as magnets when electricity flows through them.
- The commutator is a mechanical device used for switching the direction of electric current.



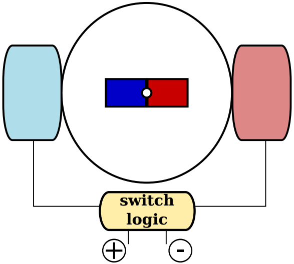
Brush Motors

- Brush motors were the predominant type of motor until the 1960s.
- The design includes three key components: the stator, the rotor and the commutator.
- Stator is typically implemented as a permanent magnet.
- Rotors are coils which act as magnets when electricity flows through them.
- The commutator is a mechanical device used for switching the direction of electric current.



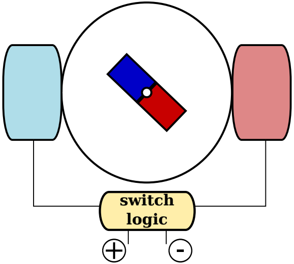
Brushless Motors

- Brush motors have a major disadvantage: the mechanical components wear off and generate sparks.
- Digital logic circuits invented in the 60s allowed for brushless commutator designs.
- Typical brushless motors use a smaller and lighter permanent magnet, suspended in a magnetic field generated by static coils.
- The commutator is typically implemented using semiconductor elements.



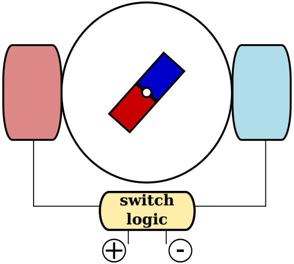
Brushless Motors

- Brush motors have a major disadvantage: the mechanical components wear off and generate sparks.
- Digital logic circuits invented in the 60s allowed for brushless commutator designs.
- Typical brushless motors use a smaller and lighter permanent magnet, suspended in a magnetic field generated by static coils.
- The commutator is typically implemented using semiconductor elements.



Brushless Motors

- Brush motors have a major disadvantage: the mechanical components wear off and generate sparks.
- Digital logic circuits invented in the 60s allowed for brushless commutator designs.
- Typical brushless motors use a smaller and lighter permanent magnet, suspended in a magnetic field generated by static coils.
- The commutator is typically implemented using semiconductor elements.



Stepper Motors and Servos

- Motors in robotics and automation applications often have to be actuated to achieve particular rotational increments.
- Most popular designs for this application are stepper motors and servos.
- Stepper motors are brushless motors with a cogwheel-like tooth design.
- By using a higher number of stators, stepper motors can be commanded to actuate one tooth at a time and can achieve and hold desired number of rotational ticks.
- Servos are typically implemented as a brushless motor, coupled with a rotary encoder and a microcontroller. We will discuss servos more in the lab.

Stepper Motors and Servos

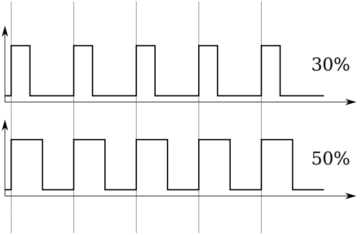
- Motors in robotics and automation applications often have to be actuated to achieve particular rotational increments.
- Most popular designs for this application are stepper motors and servos.
- Stepper motors are brushless motors with a cogwheel-like tooth design.
- By using a higher number of stators, stepper motors can be commanded to actuate one tooth at a time and can achieve and hold desired number of rotational ticks.
- Servos are typically implemented as a brushless motor, coupled with a rotary encoder and a microcontroller. We will discuss servos more in the lab.

Stepper Motors and Servos

- Motors in robotics and automation applications often have to be actuated to achieve particular rotational increments.
- Most popular designs for this application are stepper motors and servos.
- Stepper motors are brushless motors with a cogwheel-like tooth design.
- By using a higher number of stators, stepper motors can be commanded to actuate one tooth at a time and can achieve and hold desired number of rotational ticks.
- Servos are typically implemented as a brushless motor, coupled with a rotary encoder and a microcontroller. We will discuss servos more in the lab.

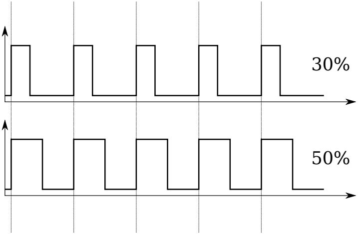
Actuating Motors: Pulse Width Modulation

- DC motors are actuated with a fixed input voltage.
- In order to move slower or faster, we could change the set input voltage (between 0 and V_{max})
- In digital circuits it is typically easier to use pulse width modulation to achieve control values.
- Wider pulses integrate to larger area under the curve and longer duty cycles. This is also safer for the motor.



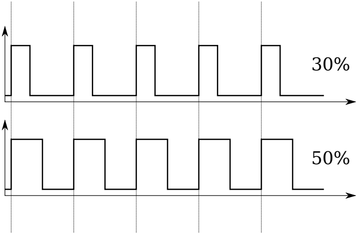
Actuating Motors: Pulse Width Modulation

- DC motors are actuated with a fixed input voltage.
- In order to move slower or faster, we could change the set input voltage (between 0 and V_{max})
- In digital circuits it is typically easier to use pulse width modulation to achieve control values.
- Wider pulses integrate to larger area under the curve and longer duty cycles. This is also safer for the motor.



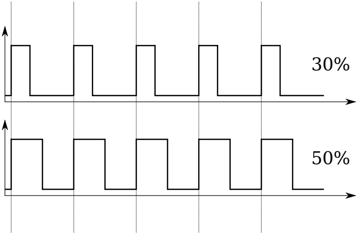
Actuating Motors: Pulse Width Modulation

- DC motors are actuated with a fixed input voltage.
- In order to move slower or faster, we could change the set input voltage (between 0 and V_{max})
- In digital circuits it is typically easier to use pulse width modulation to achieve control values.
- Wider pulses integrate to larger area under the curve and longer duty cycles. This is also safer for the motor.



Actuating Motors: Pulse Width Modulation

- DC motors are actuated with a fixed input voltage.
- In order to move slower or faster, we could change the set input voltage (between 0 and V_{max})
- In digital circuits it is typically easier to use pulse width modulation to achieve control values.
- Wider pulses integrate to larger area under the curve and longer duty cycles. This is also safer for the motor.



Torque, RPM and Gears

- In some applications the motor output is directly linked to the driven mechanical system (direct drive).
- However, it is also common to drive output through a system of gears.

Torque, RPM and Gears

- In some applications the motor output is directly linked to the driven mechanical system (direct drive).
- However, it is also common to drive output through a system of gears.

photo from www.emerson-ept.com

Torque, RPM and Gears

- Given two gears A and B with radius r_a and r_b, the contact point moves at the same tangetial speed:

$$v = r_a \omega_a = r_b \omega_b$$

- The number of teeth on each gear n_a and n_b is proportional to the circomference, and thus to the radius of the gear.
- Therefore:

$$\frac{\omega_a}{\omega_b} = \frac{n_b}{n_a}$$

- Conversely, the exertable torques τ_a and τ_b are proportional to the arm lengths r_a, r_b, and thus:

$$\frac{\tau_b}{\tau_a} = \frac{n_b}{n_a}$$

Torque, RPM and Gears

- Given two gears A and B with radius r_a and r_b, the contact point moves at the same tangetial speed:

$$v = r_a\omega_a = r_b\omega_b$$

- The number of teeth on each gear n_a and n_b is proportional to the circomference, and thus to the radius of the gear.
- Therefore:

$$\frac{\omega_a}{\omega_b} = \frac{n_b}{n_a}$$

- Conversely, the exertable torques τ_a and τ_b are proportional to the arm lengths r_a, r_b, and thus:

$$\frac{\tau_b}{\tau_a} = \frac{n_b}{n_a}$$

Torque, RPM and Gears

- Given two gears A and B with radius r_a and r_b, the contact point moves at the same tangetial speed:

$$v = r_a \omega_a = r_b \omega_b$$

- The number of teeth on each gear n_a and n_b is proportional to the circomference, and thus to the radius of the gear.
- Therefore:

$$\frac{\omega_a}{\omega_b} = \frac{n_b}{n_a}$$

- Conversely, the exertable torques τ_a and τ_b are proportional to the arm lengths r_a, r_b, and thus:

$$\frac{\tau_b}{\tau_a} = \frac{n_b}{n_a}$$

Torque, RPM and Gears

- Given two gears A and B with radius r_a and r_b, the contact point moves at the same tangetial speed:

$$v = r_a \omega_a = r_b \omega_b$$

- The number of teeth on each gear n_a and n_b is proportional to the circomference, and thus to the radius of the gear.
- Therefore:

$$\frac{\omega_a}{\omega_b} = \frac{n_b}{n_a}$$

- Conversely, the exertable torques τ_a and τ_b are proportional to the arm lengths r_a, r_b, and thus:

$$\frac{\tau_b}{\tau_a} = \frac{n_b}{n_a}$$

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Torque, RPM and Gears

- A high input to output gear ratio ($n_a > n_b$) reduces the exerted torque and increases the angular velocity.
- More commonly, gear trains with a higher number of output teeth are used and the gear ratio is reported as $n_b : 1$.
- Gear systems are characterized also by:
  - efficiency: the ratio of useful torque transmitted (some torque is always lost due to friction and inertia)
  - backclash: how much the output shaft can be rotated back before the input shaft moves
  - backdrivability: can rotations on the output shaft drive back the motor.

Motor Data Sheets

Parameters of motors to check when designing your application:

- Operating voltage
- Maximum load current
- Maximum short-term current
- Maximum torque
- Maximum rpm
- Gear ratio

Outline

Encoder Basics

- The rotary encoder is one of the basic sensors in robotics
- A rotary encoder is typically mounted on a motor shaft or gear output shaft and used to count the numbers (and fractions) of rotation.
- Encoder output is typically used as feedback for motor control in servo systems.
- Wheel encoder readings are used in vehicle kinematic models to deduce the relative position and orientation of mobile robots (odometry).

Encoder Basics

- The rotary encoder is one of the basic sensors in robotics
- A rotary encoder is typically mounted on a motor shaft or gear output shaft and used to count the numbers (and fractions) of rotation.
- Encoder output is typically used as feedback for motor control in servo systems.
- Wheel encoder readings are used in vehicle kinematic models to deduce the relative position and orientation of mobile robots (odometry).

Encoder Basics

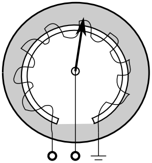
- The rotary encoder is one of the basic sensors in robotics
- A rotary encoder is typically mounted on a motor shaft or gear output shaft and used to count the numbers (and fractions) of rotation.
- Encoder output is typically used as feedback for motor control in servo systems.
- Wheel encoder readings are used in vehicle kinematic models to deduce the relative position and orientation of mobile robots (odometry).

Encoder Basics

- The rotary encoder is one of the basic sensors in robotics
- A rotary encoder is typically mounted on a motor shaft or gear output shaft and used to count the numbers (and fractions) of rotation.
- Encoder output is typically used as feedback for motor control in servo systems.
- Wheel encoder readings are used in vehicle kinematic models to deduce the relative position and orientation of mobile robots (odometry).

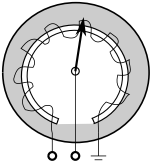
Potentiometers

- A potentiometer consists of a resistor and a movable contact element.
- When the contact element moves, it divides the resistor in to two resistance elements R_1 and R_2.
- The terminals are connected over a load resistor R_L and the voltage over R_L is calculated as $V_L = \frac{R_1}{R_1+R_2} V_{in}$
- This is a typical voltage divider circuit. Knowing V_{in} and $R_1 + R_2$, we can deduce the position of the contact element from the measured voltage V_L.



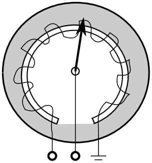
Potentiometers

- A potentiometer consists of a resistor and a movable contact element.
- When the contact element moves, it divides the resistor in to two resistance elements R_1 and R_2.
- The terminals are connected over a load resistor R_L and the voltage over R_L is calculated as $V_L = \frac{R_1}{R_1+R_2} V_{in}$
- This is a typical voltage divider circuit. Knowing V_{in} and $R_1 + R_2$, we can deduce the position of the contact element from the measured voltage V_L.



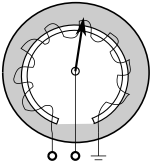
Potentiometers

- A potentiometer consists of a resistor and a movable contact element.
- When the contact element moves, it divides the resistor in to two resistance elements R_1 and R_2.
- The terminals are connected over a load resistor R_L and the voltage over R_L is calculated as $V_L = \frac{R_1}{R_1+R_2} V_{in}$
- This is a typical voltage divider circuit. Knowing V_{in} and $R_1 + R_2$, we can deduce the position of the contact element from the measured voltage V_L.



Potentiometers

- A potentiometer consists of a resistor and a movable contact element.
- When the contact element moves, it divides the resistor in to two resistance elements R_1 and R_2.
- The terminals are connected over a load resistor R_L and the voltage over R_L is calculated as $V_L = \frac{R_1}{R_1+R_2}V_{in}$
- This is a typical voltage divider circuit. Knowing V_{in} and $R_1 + R_2$, we can deduce the position of the contact element from the measured voltage V_L.

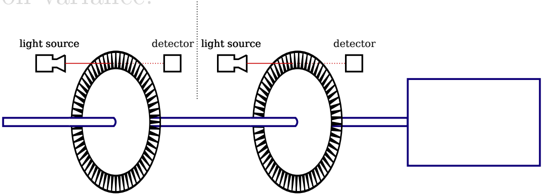


Optical Encoders: Incremental

- Potentiometers don’t allow for continuous rotation.
- Optical encoders can provide continuous rotation measurements using light sensitive elements and an encoded disk.

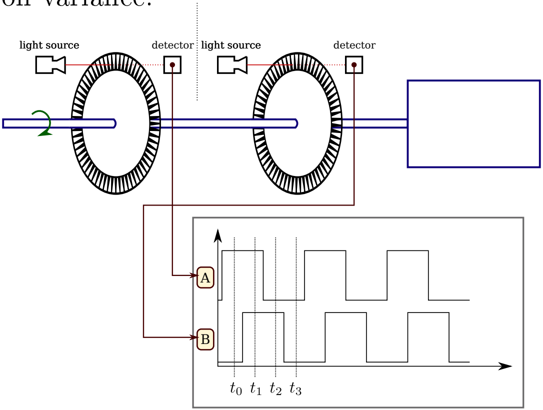
Optical Encoders: Incremental

- Incremental optical encoders use one disk with equally spaced black lines to generate multiple signal pulses.
- A second disk is added at a 90° phase offset to add for direction variance.



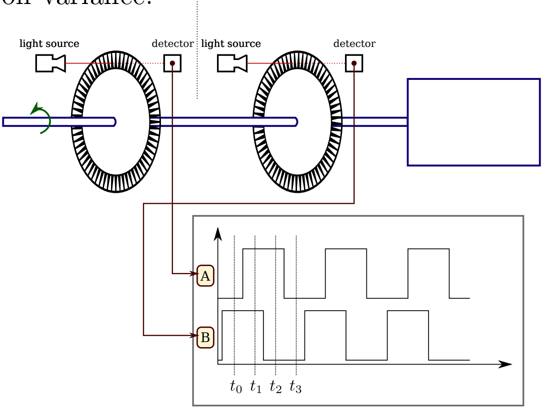
Optical Encoders: Incremental

- Incremental optical encoders use one disk with equally spaced black lines to generate multiple signal pulses.
- A second disk is added at a 90° phase offset to add for direction variance.



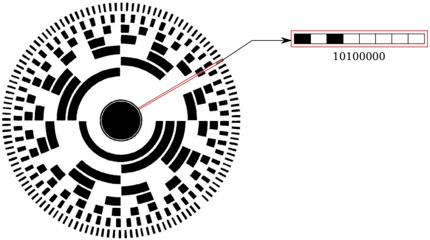
Optical Encoders: Incremental

- Incremental optical encoders use one disk with equally spaced black lines to generate multiple signal pulses.
- A second disk is added at a 90^o phase offset to add for direction variance.



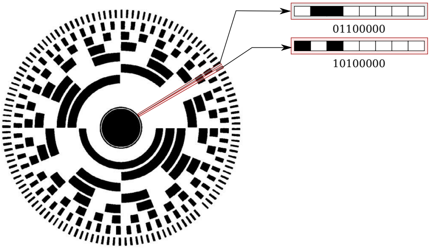
Optical Encoders: Absolute

- Absolute encoders solve the problem of initialization errors in incremental encoders.
- Each tick of the encoder has a unique optical bit word.
- With gray codes neighboring words have only 1 different bit, adding robustness to measurement error.



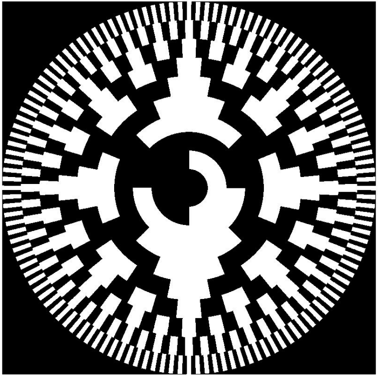
Optical Encoders: Absolute

- Absolute encoders solve the problem of initialization errors in incremental encoders.
- Each tick of the encoder has a unique optical bit word.
- With gray codes neighboring words have only 1 different bit, adding robustness to measurement error.



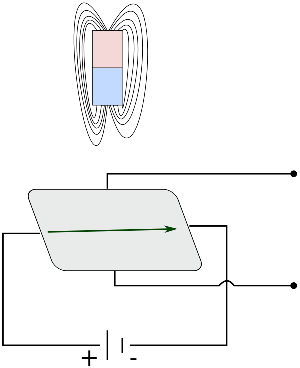
Optical Encoders: Absolute

- Absolute encoders solve the problem of initialization errors in incremental encoders.
- Each tick of the encoder has a unique optical bit word.
- With gray codes neighboring words have only 1 different bit, adding robustness to measurement error.



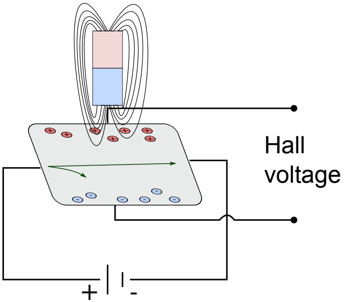
Magnetic Encoders

- Sensors based on magnetic field strength can also be used to implement rotary encoders.
- Typical magnetic encoders use the Hall effect which produces voltage potential proportional to the proximity of a magnet.
- A small disc permanent magnet is fixed on the rotating shaft, while four hall sensors provide 360 degree sensitivity.



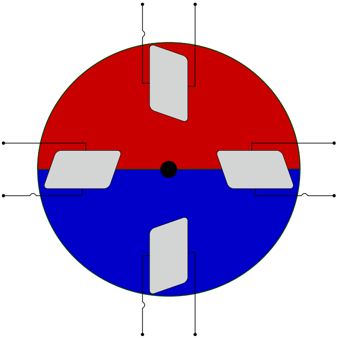
Magnetic Encoders

- Sensors based on magnetic field strength can also be used to implement rotary encoders.
- Typical magnetic encoders use the Hall effect which produces voltage potential proportional to the proximity of a magnet.
- A small disc permanent magnet is fixed on the rotating shaft, while four hall sensors provide 360 degree sensitivity.



Magnetic Encoders

- Sensors based on magnetic field strength can also be used to implement rotary encoders.
- Typical magnetic encoders use the Hall effect which produces voltage potential proportional to the proximity of a magnet.
- A small disc permanent magnet is fixed on the rotating shaft, while four hall sensors provide 360 degree sensitivity.



Interfacing Encoders

- Encoders are often packaged with their own microcontrollers for low-level logic. Reading encoder measurements from those is device-dependent and follows manufacturer-defined protocols.
- Occasionally, you may need to read encoder ticks directly in your microcontroller code.
- The encoder square waves are often tied to interrupts on the microprocessor. Once an interrupt is triggered, you may need to check other input pins to determine direction of the wave, before updating encoder positions incrementally.
- You will get some practical experience with this in the lab.

Interfacing Encoders

- Encoders are often packaged with their own microcontrollers for low-level logic. Reading encoder measurements from those is device-dependent and follows manufacturer-defined protocols.
- Occasionally, you may need to read encoder ticks directly in your microcontroller code.
- The encoder square waves are often tied to interrupts on the microprocessor. Once an interrupt is triggered, you may need to check other input pins to determine direction of the wave, before updating encoder positions incrementally.
- You will get some practical experience with this in the lab.

Interfacing Encoders

- Encoders are often packaged with their own microcontrollers for low-level logic. Reading encoder measurements from those is device-dependent and follows manufacturer-defined protocols.
- Occasionally, you may need to read encoder ticks directly in your microcontroller code.
- The encoder square waves are often tied to interrupts on the microprocessor. Once an interrupt is triggered, you may need to check other input pins to determine direction of the wave, before updating encoder positions incrementally.
- You will get some practical experience with this in the lab.

Interfacing Encoders

- Encoders are often packaged with their own microcontrollers for low-level logic. Reading encoder measurements from those is device-dependent and follows manufacturer-defined protocols.
- Occasionally, you may need to read encoder ticks directly in your microcontroller code.
- The encoder square waves are often tied to interrupts on the microprocessor. Once an interrupt is triggered, you may need to check other input pins to determine direction of the wave, before updating encoder positions incrementally.
- You will get some practical experience with this in the lab.

Outline

Open-loop Motor Actuation

- Open loop control is generally a very bad idea in precision applications
- It may however make sense if you just need to move for a pre-defined time at a pre-defined rate: just set the motor duty cycle to a specific value and hope for the best
- Motor control however implies you need to have some more sophisticated feedback: usually in terms of encoder ticks, but also load current, force or torque sensors.
- The most common control framework for simple motor control is the PID controller.

Open-loop Motor Actuation

- Open loop control is generally a very bad idea in precision applications
- It may however make sense if you just need to move for a pre-defined time at a pre-defined rate: just set the motor duty cycle to a specific value and hope for the best
- Motor control however implies you need to have some more sophisticated feedback: usually in terms of encoder ticks, but also load current, force or torque sensors.
- The most common control framework for simple motor control is the PID controller.

Open-loop Motor Actuation

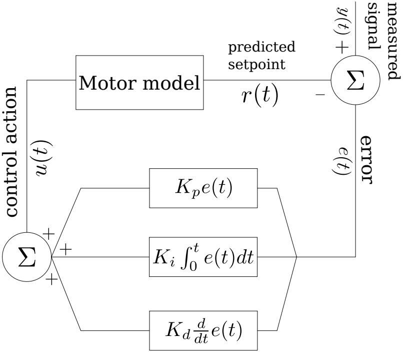
- Open loop control is generally a very bad idea in precision applications
- It may however make sense if you just need to move for a pre-defined time at a pre-defined rate: just set the motor duty cycle to a specific value and hope for the best
- Motor control however implies you need to have some more sophisticated feedback: usually in terms of encoder ticks, but also load current, force or torque sensors.
- The most common control framework for simple motor control is the PID controller.

Open-loop Motor Actuation

- Open loop control is generally a very bad idea in precision applications
- It may however make sense if you just need to move for a pre-defined time at a pre-defined rate: just set the motor duty cycle to a specific value and hope for the best
- Motor control however implies you need to have some more sophisticated feedback: usually in terms of encoder ticks, but also load current, force or torque sensors.
- The most common control framework for simple motor control is the PID controller.

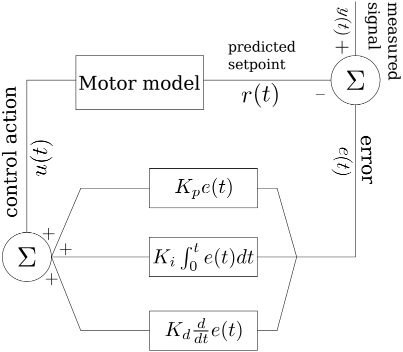
PID Control

- The Proportional Integral Derivative (PID) control framework scales control outputs in accordance with the error, it's derivative and integral.
- Given a function, which generates set point values r(t), the objective of the controller is to produce control actions u(t), such that the error e(t) = y(t) − r(t) is minimized.



PID Control

- The Proportional Integral Derivative (PID) control framework scales control outputs in accordance with the error, it's derivative and integral.
- Given a function, which generates set point values r(t), the objective of the controller is to produce control actions u(t), such that the error $e(t) = y(t) - r(t)$ is minimized.



PID Control

Proportional term:

- The proportional term applies a control, directly proportional to the current error $e(t)$
- The control action is then $u(t) = K_p e(t)$
- Pure P-control suffers from overshoots and oscillations around the steady state, as well as steady state errors.

PID Control

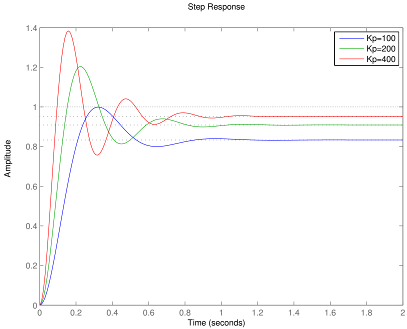
Proportional term:

- The proportional term applies a control, directly proportional to the current error $e(t)$
- The control action is then $u(t) = K_p e(t)$
- Pure P-control suffers from overshoots and oscillations around the steady state, as well as steady state errors.

PID Control

Proportional term:

- The proportional term applies a control, directly proportional to the current error $e(t)$
- The control action is then $u(t) = K_p e(t)$
- Pure P-control suffers from overshoots and oscillations around the steady state, as well as steady state errors.



PID Control

Integral term:

- The integral term applies a control, proportional to the sum of all previous errors.
- The integral control action is then $u(t) = K_i \int_0^t e(t)dt$.
- Integral terms decrease the steady state error, as they accumulate even small errors over time to generate high enough controls. Typically used in combination with a P-term.

PID Control

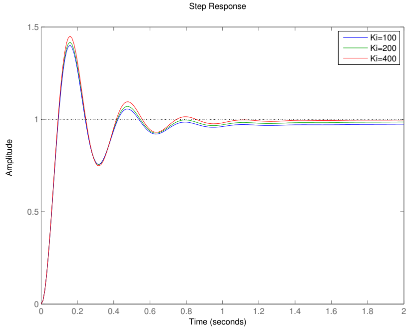
Integral term:

- The integral term applies a control, proportional to the sum of all previous errors.
- The integral control action is then $u(t) = K_i \int_0^t e(t)dt$.
- Integral terms decrease the steady state error, as they accumulate even small errors over time to generate high enough controls. Typically used in combination with a P-term.

PID Control

Integral term:

- The integral term applies a control, proportional to the sum of all previous errors.
- The integral control action is then $u(t) = K_i \int_0^t e(t)dt$.
- Integral terms decrease the steady state error, as they accumulate even small errors over time to generate high enough controls. Typically used in combination with a P-term.



PID Control

Derivative term:

- The derivative term applies controls, proportional to the derivative of the error.
- The derivative control action is then $u(t) = K_d \frac{d}{dt}e(t)$.
- Derivative terms dampen oscillations and act to smoothen the system behavior. Typically, used in combination with a P-term.

PID Control

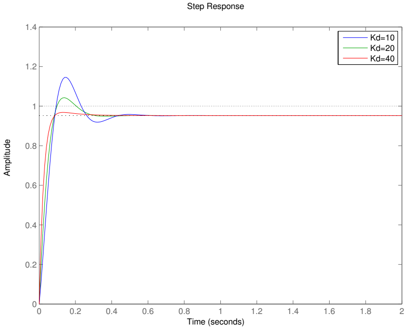
Derivative term:

- The derivative term applies controls, proportional to the derivative of the error.
- The derivative control action is then $u(t) = K_d \frac{d}{dt} e(t)$.
- Derivative terms dampen oscillations and act to smoothen the system behavior. Typically, used in combination with a P-term.

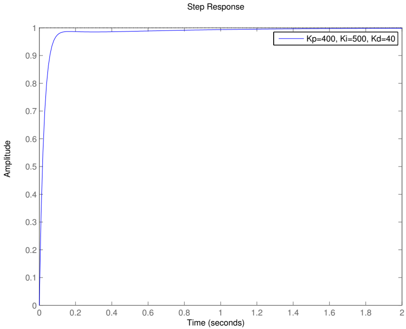
PID Control

Derivative term:

- The derivative term applies controls, proportional to the derivative of the error.
- The derivative control action is then $u(t) = K_d \frac{d}{dt} e(t)$.
- Derivative terms dampen oscillations and act to smoothen the system behavior. Typically, used in combination with a P-term.



PID Control



The full PID equation is then:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{1}$$

Tuning of PID Gains

Gain	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	No effect

Tuning of PID Gains

Gain	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	No effect

- Start by tuning K_p. Increase until you see steady state oscillations.
- Add a K_d term to decrease overshoot.
- If there is a steady state error, add a small K_i to eliminate it.

Tuning of PID Gains

Gain	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	No effect

- Start by tuning K_p. Increase until you see steady state oscillations.
- Add a K_d term to decrease overshoot.
- If there is a steady state error, add a small K_i to eliminate it.

Tuning of PID Gains

Gain	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	No effect

- Start by tuning K_p. Increase until you see steady state oscillations.
- Add a K_d term to decrease overshoot.
- If there is a steady state error, add a small K_i to eliminate it.

Common Modes of Motor Control

- Position control: use an encoder after the gears to feedback the position for control.
- Velocity control: use the first time derivative of the encoder position as a control variable.
- Current control: use the electrical current absorption for control.
- Force/Torque control: couple the motor with a force or torque sensor and use the output in control space. Common mode for achieving compliance.

Common Modes of Motor Control

- Position control: use an encoder after the gears to feedback the position for control.
- Velocity control: use the first time derivative of the encoder position as a control variable.
- Current control: use the electrical current absorption for control.
- Force/Torque control: couple the motor with a force or torque sensor and use the output in control space. Common mode for achieving compliance.

Common Modes of Motor Control

- Position control: use an encoder after the gears to feedback the position for control.
- Velocity control: use the first time derivative of the encoder position as a control variable.
- Current control: use the electrical current absorption for control.
- Force/Torque control: couple the motor with a force or torque sensor and use the output in control space. Common mode for achieving compliance.

Common Modes of Motor Control

- Position control: use an encoder after the gears to feedback the position for control.
- Velocity control: use the first time derivative of the encoder position as a control variable.
- Current control: use the electrical current absorption for control.
- Force/Torque control: couple the motor with a force or torque sensor and use the output in control space. Common mode for achieving compliance.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor x(t) (offset from a reference angle in radians)
- Non-smooth x(t) may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of x(t) vanishes.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor $x(t)$ (offset from a reference angle in radians)
- Non-smooth $x(t)$ may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of $x(t)$ vanishes.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor $x(t)$ (offset from a reference angle in radians)
- Non-smooth $x(t)$ may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of $x(t)$ vanishes.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor $x(t)$ (offset from a reference angle in radians)
- Non-smooth $x(t)$ may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of $x(t)$ vanishes.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor x(t) (offset from a reference angle in radians)
- Non-smooth x(t) may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of x(t) vanishes.

Generating smooth motion

- It is often desireable to generate a smooth trajectory for the driven system (manipulator or wheeled base).
- Given the position of a motor $x(t)$ (offset from a reference angle in radians)
- Non-smooth $x(t)$ may require an infinite velocity $x'(t) = \frac{d}{dt}x(t)$
- Similarly, the velocity needs to be smooth, or else we risk infinite acceleration $x''(t)$
- Ideally, we want a smoothly varying acceleration profile, thus we want to minimize the integral of the jerk $x'''(t)$ over the motion.
- Theoretically[1], this occurs when the sixth derivative of $x(t)$ vanishes.

Generating smooth motion

We represent $x(t)$ as a fifth order polynomial:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{2}$$

With derivatives:

$$x'(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \tag{3}$$

$$x''(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \tag{4}$$

Assuming we start at rest at 0 and want to finish at rest at d in time τ, we know the boundary conditions:

$$x(0) = 0, \quad x(\tau) = d \tag{5}$$

$$x'(0) = 0, \quad x'(\tau) = 0 \tag{6}$$

$$x''(0) = 0, \quad x''(\tau) = 0 \tag{7}$$

Generating smooth motion

- This leaves us with 6 equations for 6 unknowns, with the solution:

$$x(t) = d\left(10\left(\frac{t}{\tau}\right)^3 - 15\left(\frac{t}{\tau}\right)^4 + 6\left(\frac{t}{\tau}\right)^5\right) \qquad (8)$$

- Why minimum jerk?
- -> keeps the maximum velocity and acceleration within reasonable bounds.
- In practice, reduces the size of the step sent to the PID controllers.

Generating smooth motion

- This leaves us with 6 equations for 6 unknowns, with the solution:

$$x(t) = d\left(10\left(\frac{t}{\tau}\right)^3 - 15\left(\frac{t}{\tau}\right)^4 + 6\left(\frac{t}{\tau}\right)^5\right) \tag{8}$$

- Why minimum jerk?
- -> keeps the maximum velocity and acceleration within reasonable bounds.
- In practice, reduces the size of the step sent to the PID controllers.

Generating smooth motion

- This leaves us with 6 equations for 6 unknowns, with the solution:

$$x(t) = d\left(10\left(\frac{t}{\tau}\right)^3 - 15\left(\frac{t}{\tau}\right)^4 + 6\left(\frac{t}{\tau}\right)^5\right) \tag{8}$$

- Why minimum jerk?
- -> keeps the maximum velocity and acceleration within reasonable bounds.
- In practice, reduces the size of the step sent to the PID controllers.

Generating smooth motion

- This leaves us with 6 equations for 6 unknowns, with the solution:

$$x(t) = d\left(10\left(\frac{t}{\tau}\right)^3 - 15\left(\frac{t}{\tau}\right)^4 + 6\left(\frac{t}{\tau}\right)^5\right) \quad (8)$$

- Why minimum jerk?
- -> keeps the maximum velocity and acceleration within reasonable bounds.
- In practice, reduces the size of the step sent to the PID controllers.

References

Neville Hogan.
An organizing principle for a class of voluntary movements.
The Journal of Neuroscience, 4(11):2745–2754, 1984.

Sensors and Sensing
Motors, Encoders and Control

Todor Stoyanov

Centre for Applied Autonomous Sensor Systems
Örebro University
Sweden