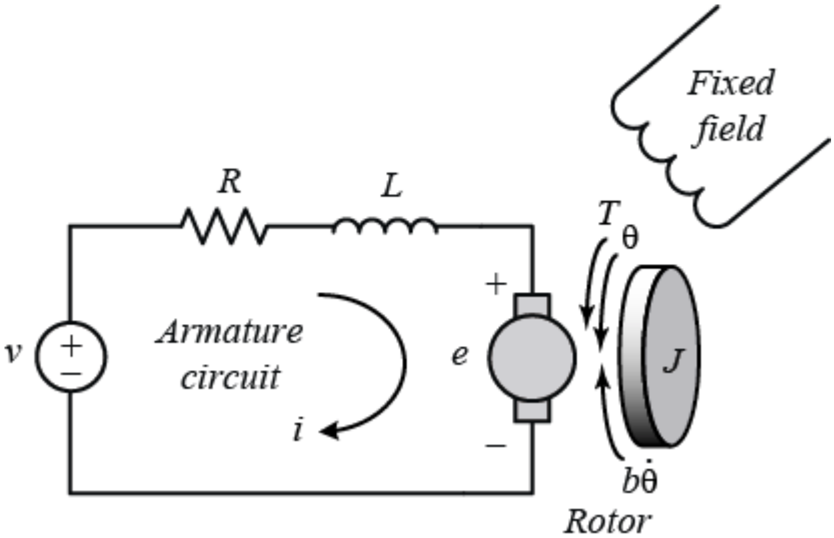
Control Tutorials for MATLAB and Simulink

Key MATLAB commands used in this tutorial are: `tf` , `ss`

Contents

Physical setup

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion. The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure.



For this example, we will assume that the input of the system is the voltage source (V) applied to the motor's armature, while the output is the rotational speed of the shaft $\dot{\theta}$. The rotor and shaft are assumed to be rigid. We further assume a viscous friction model, that is, the friction torque is proportional to shaft angular velocity.

The physical parameters for our example are:

```
(J)     moment of inertia of the rotor     0.01 kg.m^2

(b)     motor viscous friction constant    0.1 N.m.s

(Ke)    electromotive force constant       0.01 V/rad/sec

(Kt)    motor torque constant              0.01 N.m/Amp

(R)     electric resistance                1 Ohm

(L)     electric inductance                0.5 H
```

System equations

In general, the torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torque is proportional to only the armature current i by a constant factor K_t as shown in the equation below. This is referred to as an armature-controlled motor.

(1) $T = K_t i$

The back emf, e, is proportional to the angular velocity of the shaft by a constant factor K_e.

(2) $e = K_e \dot{\theta}$

In SI units, the motor torque and back emf constants are equal, that is, $K_t = K_e$; therefore, we will use K to represent both the motor torque constant and the back emf constant.

From the figure above, we can derive the following governing equations based on Newton's 2nd law and Kirchhoff's voltage law.

(3) $J\ddot{\theta} + b\dot{\theta} = Ki$

(4) $L\frac{di}{dt} + Ri = V - K\dot{\theta}$

1. Transfer Function

Applying the Laplace transform, the above modeling equations can be expressed in terms of the Laplace variable *s*.

(5) $s(Js + b)\Theta(s) = KI(s)$

(6) $(Ls + R)I(s) = V(s) - Ks\Theta(s)$

We arrive at the following open-loop transfer function by eliminating $I(s)$ between the two above equations, where the rotational speed is considered the output and the armature voltage is considered the input.

(7) $P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \qquad [\frac{rad/sec}{V}]$

2. State-Space

In state-space form, the governing equations above can be expressed by choosing the rotational speed and electric current as the state variables. Again the armature voltage is treated as the input and the rotational speed is chosen as the output.

(8) $\frac{d}{dt}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}$$
(9)

Design requirements

First consider that our uncompensated motor rotates at 0.1 rad/sec in steady state for an input voltage of 1 Volt (this is demonstrated in the DC Motor Speed: System Analysis page where the system's open-loop response is simulated). Since the most basic requirement of a motor is that it should rotate at the desired speed, we will require that the steady-state error of the motor speed be less than 1%. Another performance requirement for our motor is that it must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time less than 2 seconds. Also, since a speed faster than the reference may damage the equipment, we want to have a step response with overshoot of less than 5%.

In summary, for a unit step command in motor speed, the control system's output should meet the following requirements.

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

MATLAB representation

1. Transfer Function

We can represent the above open-loop transfer function of the motor in MATLAB by defining the parameters and transfer function as follows. Running this code in the command window produces the output shown below.

```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+K^2)
```

```
P_motor =

             0.01
  ---------------------------
  0.005 s^2 + 0.06 s + 0.1001

Continuous-time transfer function.
```

2. State Space

We can also represent the system using the state-space equations. The following additional MATLAB commands create a state-space model of the motor and produce the output shown below when run in the MATLAB command window.

```
A = [-b/J   K/J
    -K/L   -R/L];
B = [0
    1/L];
C = [1   0];
D = 0;
motor_ss = ss(A,B,C,D)
```

```
motor_ss =

  A =
          x1     x2
   x1    -10      1
   x2  -0.02     -2

  B =
       u1
   x1   0
   x2   2

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

The above state-space model can also be generated by converting your existing transfer function model into state-space form. This is again accomplished with the `ss` command as shown below.

```
motor_ss = ss(P_motor);
```