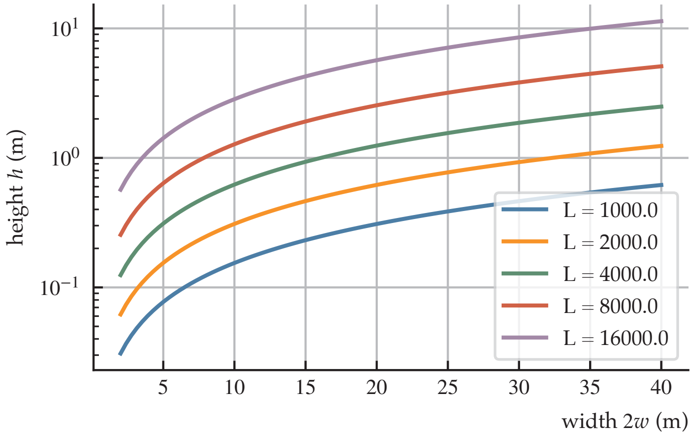


Figure S4.2. Truss design curve.

**Problem 4.6** W5 Consider an LTI system modeled by the state equation of the state-space model, equation (4.24a). A **steady state** of a system is defined as the state vector $\boldsymbol{x}(t)$ after the effects of initial conditions have become relatively small. For a constant input $\boldsymbol{u}(t) = \overline{\boldsymbol{u}}$, the constant state $\overline{\boldsymbol{x}}$ toward which the system's response decays can be found by setting the time derivative vector $\boldsymbol{x}'(t) = \boldsymbol{0}$.

Write a Python function `steady_state()` that accepts the following arguments:

- `A`: A symbolic matrix representing $A$
- `B`: A symbolic matrix representing $B$
- `u_const`: A symbolic vector representing $\overline{\boldsymbol{u}}$

The function should return `x_const`, a symbolic vector representing $\overline{\boldsymbol{x}}$.

The steady-state output converges to $\overline{\boldsymbol{y}}$ the corresponding output equation of the state-space model, equation (4.24b). Write a second Python function `steady_output()` that accepts the following arguments:

- `C`: A symbolic matrix representing $C$
- `D`: A symbolic matrix representing $D$
- `u_const`: A symbolic vector representing $\overline{\boldsymbol{u}}$
- `x_const`: A symbolic vector representing $\overline{\boldsymbol{x}}$

This function should return `y_const`, a symbolic vector representing $\overline{\boldsymbol{y}}$.

Apply `steady_state()` and `steady_output()` to the state-space model of the circuit shown in figure 4.7, which includes a resistor with resistance $R$, an inductor with inductance $L$, and capacitor with capacitance $C$. The LTI system is represented by equation (4.24) with state, input, and output vectors

$$x(t) = \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix}, \; u(t) = \begin{bmatrix} V_S \end{bmatrix}, \; y(t) = \begin{bmatrix} v_C(t) \\ v_L(t) \end{bmatrix}$$

and the following matrices:

$$A = \begin{bmatrix} 0 & 1/C \\ -1/L & -R/L \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 1/L \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 \\ -1 & -R \end{bmatrix}, \; D = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Furthermore, let the constant input vector be

$$\overline{u} = \begin{bmatrix} \overline{V_S} \end{bmatrix},$$

for constant $\overline{V_S}$.

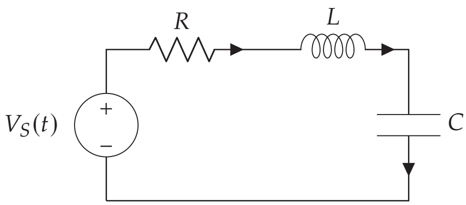


Figure 4.7. An RLC circuit with a voltage source $V_S(t)$.

**Solution 4.6** W5 A constant steady-state, $x' = \mathbf{0}$ implies, from the state equation (4.24a),

$$\mathbf{0} = A\overline{x} + B\overline{u} \Rightarrow \tag{4.9}$$

$$\overline{x} = -A^{-1}B\overline{u}. \tag{4.10}$$

We are now ready to define `steady_state()` as follows:

```
def steady_state(A, B, u_const):
  """Returns the symbolic constant steady state vector"""
  A = sp.Matrix(A)  # In case A isn't symbolic
  B = sp.Matrix(B)  # In case B isn't symbolic
  u_const = sp.Matrix(u_const)  # In case u_const isn't symbolic
  x_const = -A**-1 * B * u_const
  return x_const
```

The state-space output equation equation (4.24b) is already solved for the output, so we are ready to write `steady_output()` as follows:

```python
def steady_output(C, D, u_const, x_const):
    """Returns the symbolic constant steady-state output vector"""
    C = sp.Matrix(C)  # In case C isn't symbolic
    D = sp.Matrix(D)  # In case D isn't symbolic
    u_const = sp.Matrix(u_const)  # In case u_const isn't symbolic
    x_const = sp.Matrix(x_const)  # In case x_const isn't symbolic
    y_const = C*x_const + D*u_const
    return y_const
```

Apply these functions to the given state-space model. First, define the symbolic variables as follows:

```python
R, L, C1 = sp.symbols("R, L, C1", positive=True)
VS_ = sp.symbols("VS_", real=True)  # Constant voltage source input
```

Now define the system and the constant input as follows:

```python
A = sp.Matrix([[0, 1/C1], [-1/L, -R/L]])  # A
B = sp.Matrix([[0], [1/L]])  # B
C = sp.Matrix([[1, 0], [-1, -R]])  # C
D = sp.Matrix([[0], [1]])  # D
u_const = sp.Matrix([[VS_]])  # ū
```

Find the constant steady state $\overline{\boldsymbol{x}}$ as follows:

```python
x_const = steady_state(A, B, u_const)
print(x_const)
```

$$\begin{bmatrix} VS \\ 0 \end{bmatrix}$$

Find the constant steady-state output $\overline{\boldsymbol{y}}$ as follows:

```python
y_const = steady_output(C, D, u_const, x_const)
print(y_const)
```

$$\begin{bmatrix} VS \\ 0 \end{bmatrix}$$

**Problem 4.7** 8U Consider the electromechanical schematic of a direct current (DC) motor shown in figure 4.8. A voltage source $V_S(t)$ provides power, the armature winding loses some energy to heat through a resistance $R$ and stores some energy in a magnetic field due to its inductance $L$, which arises from its coiled structure. An electromechanical interaction through the magnetic field, shown as M, has torque constant $K_t$ and induces a torque on the motor shaft, which is supported by bearings that lose some energy to heat via a damping coefficient $B$. The rotor's mass has rotational moment of inertia $J$, which stores kinetic energy. We denote the voltage across an element with $v$, the current through an element with $i$, the angular velocity across an element with $\Omega$, and the torque through an element with $T$.

For a given input voltage and initial conditions, the following vector-valued functions have been solved for:

$$\boldsymbol{F} = \begin{bmatrix} \int_0^t v_R(t)\,dt \\ \int_0^t v_L(t)\,dt \\ \int_0^t \Omega_B(t)\,dt \\ \int_0^t \Omega_J(t)\,dt \end{bmatrix} = \begin{bmatrix} \exp(-t) \\ \exp(-t) \\ 1-\exp(-t) \\ 1-\exp(-t) \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} \int_0^t i_R(t)\,dt \\ \int_0^t i_L(t)\,dt \\ \int_0^t T_B(t)\,dt \\ \int_0^t T_J(t)\,dt \end{bmatrix} = \begin{bmatrix} \exp(-t) \\ \exp(-t) \\ 1-\exp(-t) \\ \exp(-t) \end{bmatrix}$$

The instantaneous power lossed or stored by each element is given by the following vector of products:

$$\boldsymbol{\mathcal{P}}(t) = \begin{bmatrix} v_R(t)i_R(t) \\ v_L(t)i_L(t) \\ \Omega_B(t)T_B(t) \\ \Omega_J(t)T_J(t) \end{bmatrix}.$$

The energy $\boldsymbol{\mathcal{E}}(t)$ of the elements, then, is

$$\boldsymbol{\mathcal{E}}(t) = \int_0^t \boldsymbol{\mathcal{P}}(t)dt.$$

Write a program that satisfies the following requirements:

a. It defines a function `power(F, G)` that returns the symbolic power vector $\boldsymbol{\mathcal{P}}(t)$ from any inputs $\boldsymbol{F}$ and $\boldsymbol{G}$
b. It defines a function `energy(F, G)` that returns the symbolic energy $\boldsymbol{\mathcal{E}}(t)$ from any inputs $\boldsymbol{F}$ and $\boldsymbol{G}$ (`energy()` should call `power()`)
c. It tests the `energy()` on the specific $\boldsymbol{F}$ and $\boldsymbol{G}$ given above

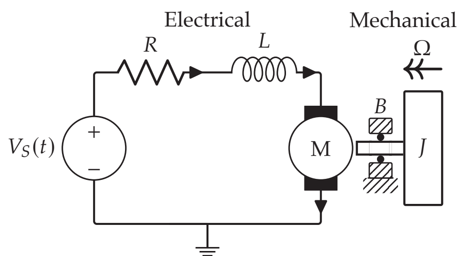


Figure 4.8. An electromechanical schematic of a DC motor.

**Solution 4.7** 8U The formula for the power of each element is given, so we are ready to define `power()` as follows:

```python
def power(F, G):
  """Returns the power for vectors F and G"""
  F = sp.Matrix(F)  # In case F isn't symbolic
  G = sp.Matrix(G)  # In case G isn't symbolic
  P = F.multiply_elementwise(G)
  # Alternative using a for loop:
  # P = sp.zeros(*F.shape)  # Initialize
  # for i, Fi in enumerate(F):
  #   P[i] = Fi * G[i]
  return P
```

The formula for the energy stored or dissipated by each element is given, so we are ready to write `energy()` as follows:

```python
def energy(F, G):
  """Returns the energy stored for vectors F and G"""
  P = power(F, G)
  E = sp.integrate(P, (t, 0, t))
  return E
```

Apply these functions to the given $\boldsymbol{F}$ and $\boldsymbol{G}$. First, define $\boldsymbol{F}$ and $\boldsymbol{G}$ as follows:

```python
t = sp.symbols("t", real=True)
F = sp.Matrix([
  [sp.exp(-t)],
  [sp.exp(-t)],
  [1 - sp.exp(-t)],
  [1 - sp.exp(-t)]
])
G = sp.Matrix([
  [sp.exp(-t)],
  [sp.exp(-t)],
  [1 - sp.exp(-t)],
  [sp.exp(-t)]
])
```

Now compute the energy:

```python
E = energy(F, G).simplify()
print(E)
```

$$\left[\begin{matrix}\frac{1}{2} - \frac{e^{-2t}}{2}\\ \frac{1}{2} - \frac{e^{-2t}}{2}\\ t - \frac{3}{2} + 2e^{-t} - \frac{e^{-2t}}{2}\\ \frac{1}{2} - e^{-t} + \frac{e^{-2t}}{2}\end{matrix}\right]$$