

CHAPTER 1: Time domain simulation

The following gives a rough overview on time domain simulation. Initially, some of the basic numerical techniques associated with the solution of ordinary differential equations is described. Next, the application of these techniques to mechanical, electrical and hydraulic systems are described.

Ordinary differential equations (ODEs) are equations that contain functions of only one independent variable and at least one term that is derived with respect to that independent variable. The ODEs focused on in this text have *time* as the independent variable. All other variables are explicit or implicit functions of time. If the term with the highest order (the term that is differentiated most often with respect to time) can be isolated, the ODE is an explicit ODE. The differential equations used to describe physical systems are often explicit ODEs. Hence, a general set of explicit ODEs may be written as:

$$\begin{aligned} \dot{\underline{q}} &= \underline{\Phi}(t, \underline{q}) \\ \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} &= \begin{bmatrix} \Phi_1(t, \underline{q}) \\ \Phi_2(t, \underline{q}) \\ \vdots \\ \Phi_n(t, \underline{q}) \end{bmatrix} \end{aligned} \quad (1.1)$$

In (1.1) n is the size of the set and the corresponding variables $\underline{q} = [q_1 \ q_2 \ \dots \ q_n]$, are the state variables. The ODE set is characterized by the n characteristic functions $\underline{\Phi} = [\Phi_1 \ \Phi_2 \ \dots \ \Phi_n]$ that, in general, can be explicit functions of time and the state variables.

The set of ODEs are solved by determining the variations of the state variables \underline{q} with time. Normally, the solution is sought for a time interval, i.e., for $t \in [0 \ T]$, where T is the size of the interval = final time. A unique solution will exist if the initial conditions are known, i.e., the values of the state variables at $t = 0$. Unfortunately, most practical systems are described by means of non-linear equations yielding governing equations that have no analytical solution. Therefore, the set of ODEs must be solved numerically. This can be done in several ways, but basically, the solution has to be carried out in a number of steps and it has to be carried out using knowledge from the previous steps, i.e., it is not possible to jump to the values at $t = T$ even though they might be the most interesting ones.

Assume that the state variables are known at a given time $t = t^{(j)}$ where the superscript indicates the j 'th step. To move forward a single time step it is necessary to predict the behavior of the state variables in the vicinity of $t^{(j)}$. This can be done by means of a Taylor expansion of \underline{q} :

$$\underline{q}^*(t) = \underline{q}^{(j)} + \frac{1}{1!} \cdot \dot{\underline{q}}^{(j)} \cdot (t - t^{(j)}) + \frac{1}{2!} \cdot \ddot{\underline{q}}^{(j)} \cdot (t - t^{(j)})^2 + \frac{1}{3!} \cdot \ddot{\underline{q}}^{(j)} \cdot (t - t^{(j)})^3 + \dots \quad (1.2)$$

In (1.2) $\underline{q}^*(t)$ approximates $\underline{q}(t)$ that is valid as long as t remains in the close vicinity of $t^{(j)}$. Assuming a sufficiently small time step (to remain in the vicinity of $t^{(j)}$) we can now compute an approximate value of $\underline{q}(t^{(j)} + h) = \underline{q}(t^{(j+1)}) = \underline{q}^{(j+1)}$:

$$\underline{q}^{(j+1)} \approx \underline{q}^*(t+h) = \underline{q}^{(j)} + \frac{1}{1!} \cdot \dot{\underline{q}}^{(j)} \cdot h + \frac{1}{2!} \cdot \ddot{\underline{q}}^{(j)} \cdot h^2 + \frac{1}{3!} \cdot \ddot{\underline{q}}^{(j)} \cdot h^3 + \dots \quad (1.3)$$

In (1.3) the time step is denoted by h . The more terms that are included in the Taylor expansion the more precise value we get for $\underline{q}^{(j+1)}$. However, for that purpose we need the derivatives of \underline{q} at the j 'th step. The first order derivatives may be determined directly from the characteristic equations, i.e.

$$\dot{\underline{q}}^{(j)} = \underline{\Phi}(t^{(j)}, \underline{q}^{(j)}) \quad (1.4)$$

Hence, a very popular and simple method is to use only the first order Taylor expansion:

$$\begin{aligned} \underline{q}^{(j+1)} &= \underline{q}^{(j)} + \dot{\underline{q}}^{(j)} \cdot h \\ \Downarrow \\ \underline{q}^{(j+1)} &= \underline{q}^{(j)} + \underline{\Phi}(t^{(j)}, \underline{q}^{(j)}) \cdot h \end{aligned} \quad (1.5)$$

This method is referred to as the forward Euler method. It is an explicit method because the new set of state variables can be computed directly. In the backward Euler method, the Taylor expansion is simply performed at the $(j+1)$ 'st step yielding:

$$\underline{q}^{(j+1)} = \underline{q}^{(j)} + \underline{\Phi}(t^{(j+1)}, \underline{q}^{(j+1)}) \cdot h \quad (1.6)$$

In this case the unknown $\underline{q}^{(j+1)}$ appears on both sides of the equation and the solution may involve an iterative solver adding to the complexity of the solution. However, this approach, will in general be more accurate.

Another explicit method is the Trapez method where an extra term of the Taylor expansion is included:

$$\underline{q}^{(j+1)} = \underline{q}^{(j)} + \dot{\underline{q}}^{(j)} \cdot h + \frac{1}{2} \cdot \ddot{\underline{q}}^{(j)} \cdot h^2 \quad (1.7)$$

An estimate of the second order derivative may be determined using information computed at the previous time step:

$$\ddot{\underline{q}}^{(j)} = \frac{1}{h} \cdot (\dot{\underline{q}}^{(j)} - \dot{\underline{q}}^{(j-1)}) \quad (1.8)$$

This yields the following expression:

$$\begin{aligned} \underline{q}^{(j+1)} &= \underline{q}^{(j)} + \frac{3}{2} \cdot \dot{\underline{q}}^{(j)} \cdot h - \frac{1}{2} \cdot \dot{\underline{q}}^{(j-1)} \cdot h \\ \Downarrow \\ \underline{q}^{(j+1)} &= \underline{q}^{(j)} + \frac{3}{2} \cdot \underline{\Phi}(t^{(j)}, \underline{q}^{(j)}) \cdot h - \frac{1}{2} \cdot \underline{\Phi}(t^{(j-1)}, \underline{q}^{(j-1)}) \cdot h \end{aligned} \quad (1.9)$$

The Trapez method can also be made implicit by estimating the second order derivative using $\dot{\underline{q}}^{(j+1)}$ in (1.8).

A somewhat more complicated explicit method is the Runge-Kutta methods that computes an estimated average of \dot{q} in the interval between $t^{(j)}$ and $t^{(j+1)}$ as a weighted average of a number of intermediate values. The most popular is the 4th order Runge-Kutta that uses four values to estimate:

$$\dot{q}(t^{(j)} \leq t < t^{(j+1)}) = \frac{1}{6} \cdot \dot{q}_A + \frac{1}{3} \cdot \dot{q}_B + \frac{1}{3} \cdot \dot{q}_C + \frac{1}{6} \cdot \dot{q}_D \quad (1.10)$$

where

$$\begin{aligned} \dot{q}_A &= \Phi(t^{(j)}, q^{(j)}) \\ \dot{q}_B &= \Phi(t^{(j)} + \frac{h}{2}, q^{(j)} + \dot{q}_A \cdot \frac{h}{2}) \\ \dot{q}_C &= \Phi(t^{(j)} + \frac{h}{2}, q^{(j)} + \dot{q}_B \cdot \frac{h}{2}) \\ \dot{q}_D &= \Phi(t^{(j)} + h, q^{(j)} + \dot{q}_C \cdot h) \end{aligned} \quad (1.11)$$

This yields the following expression for the state variables at the $j+1$ 'th:

$$q^{(j+1)} = q^{(j)} + \left(\frac{1}{6} \cdot \dot{q}_A + \frac{1}{3} \cdot \dot{q}_B + \frac{1}{3} \cdot \dot{q}_C + \frac{1}{6} \cdot \dot{q}_D \right) \cdot h \quad (1.12)$$

In general, the computation of the value of the state variables at the next time step is called time integration. There exist a wide variety of time integrations schemes. The simplest is the forward Euler, however, it requires small time steps to be sufficiently accurate. Commercial software that solve ODEs may have several different time integration schemes, often implicit ones, that are well suited to handle complex physical systems subjected to greatly varying conditions.

The numerical solution to a set of ODEs describing a physical system is called time domain simulation or, simply, simulation. Such a simulation may be thought of as an experiment carried out on the computer using a model of the physical system.

The time domain simulation can be done in the following steps:

1. Identify the size of the problem, n .
2. Set up all characteristic functions, $\Phi(t, q)$.
3. Determine initial conditions for all state variables, $q(t=0) = q^{(1)} = [q_1^{(1)} \quad q_2^{(1)} \quad \dots \quad q_n^{(1)}]$.
4. Set the step counter to 1, $j=1$.
5. Solve for the time derivatives, $\dot{q}^{(j)} = \Phi(t^{(j)}, q^{(j)})$.
6. Update state variables according to time integration scheme, e.g., $q^{(j+1)} = q^{(j)} + \dot{q}^{(j)} \cdot h$.
7. Check for saturation phenomena, and adjust state variables accordingly.
8. Update time, $t = t^{(j+1)} = t^{(j)} + h$.
9. If $t \geq T$ then the simulation (numerical solution of the ODEs) is concluded otherwise update the step counter, $j=j+1$ and return to step 5.

Clearly, computer implementation is necessary to perform these steps, especially, since steps 5..8 must be carried out for each time step. To keep computational time down the time step should be as

large as possible, however, it should not be so large that the accuracy of the simulation results become unacceptable.

Example 1

Let us consider a simple ideal mechanical pendulum, as shown in Fig. 1.1.

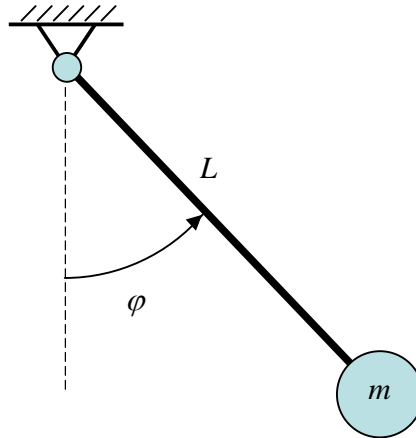


Figure 1.1 Simple pendulum with concentrated mass.

There is a single degree of freedom, φ , and the governing differential equation for this pendulum is:

$$\ddot{\varphi} = -\frac{g}{L} \cdot \sin \varphi \quad (1.13)$$

Since we have a single degree of freedom described with a second order system, the size of the problem will be $n = 2$. The state variables are:

$$\underline{q} = [q_1 \quad q_2] = [\varphi \quad \dot{\varphi}] \quad (1.14)$$

$$\underline{\dot{q}} = [\dot{q}_1 \quad \dot{q}_2] = [\dot{\varphi} \quad \ddot{\varphi}] \quad (1.15)$$

The characteristic functions are:

$$\dot{q}_1 = q_2 = \Phi_1 \quad (1.16)$$

$$\dot{q}_2 = -\frac{g}{L} \cdot \sin q_1 = \Phi_2 \quad (1.17)$$

In order to simulate this system, the required initial conditions would be:

$$\varphi(t=0) \quad \dot{\varphi}(t=0) \quad (1.18)$$

In the following an example of a program (in Matlab code) is given that solves the above problem for the pendulum starting in horizontal position at rest corresponding to the initial conditions:

$$\varphi(t=0) = \frac{\pi}{2} \quad \dot{\varphi}(t=0) = 0 \quad (1.19)$$

```
close all;
clear;
%Numerical time integration of pendulum
%Starting from rest horizontal
```

```

%The pendulum has a concentrated mass
%The pendulum has a massless length L
%from Time=0 to Time=EndTime
%Initial conditions
%phi(t=0)=90 deg = pi/2 rad
%phiDot(t=0)=0
%Basic data
g=9.81;
L=1.0;
EndTime=10.0;
StepTime=0.002;
phi_Initial=pi/2;
phiDot_Initial=0.0;
%Initialize
%Current values of phi, phiDot and phiDotDot for Time = 0
%Initially old values are simply set to current values
Time=0.0;
phi=phi_Initial;
phiDot=phiDot_Initial;
Counter=1;
%Start time integration
while Time<EndTime
    %Solve for acceleration
    phiDotDot=-g/L*sin(phi);
    %report
    Time_Plot(Counter)=Time;
    phi_Plot(Counter)=phi*180/pi;
    phiDot_Plot(Counter)=phiDot;
    phiDotDot_Plot(Counter)=phiDotDot;
    %integrate state variables
    phi=phi+phiDot*StepTime;
    phiDot=phiDot+phiDotDot*StepTime;
    Time=Time+StepTime;
    Counter=Counter+1;
end;
plot(Time_Plot,phiDot_Plot);
grid;
figure;
plot(Time_Plot,phi_Plot);
grid;

```

Executing this code yields the curves shown in Fig. 1.2

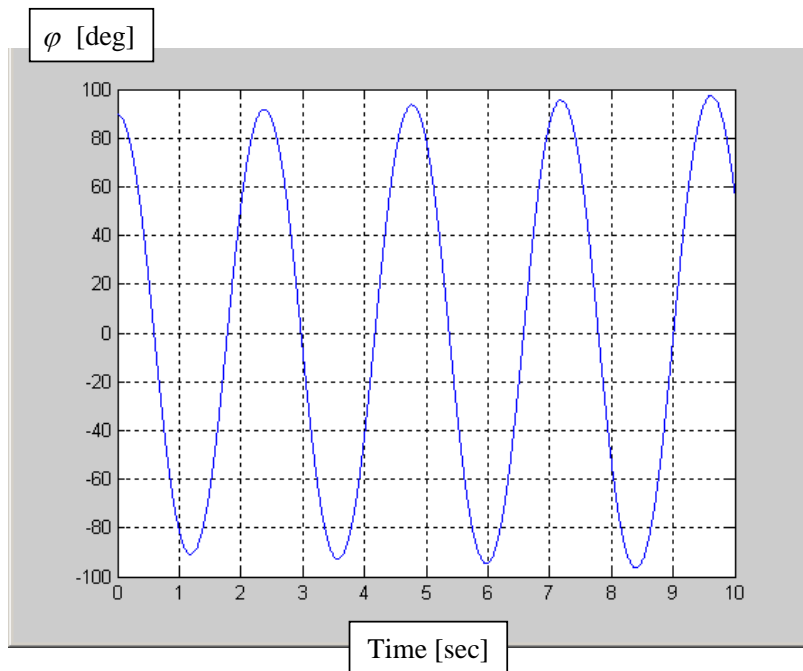


Figure 1.2 Pendulum rotation vs. time.

As can be seen from the code, a simple Forward Euler integration method is used. Clearly, an error is accumulated as the rotation of the pendulum increasingly goes beyond the ± 90 degrees. This error could be reduced by either reducing the step time or choosing a different time integration method such as Trapez or Runge-Kutta.

CHAPTER 2: One-Dimensional Mechanical Systems

The governing dynamic equations for a translating body is:

$$m \cdot \ddot{x} = \Sigma F \quad (2.1)$$

In (2.1) m is the mass and x is the coordinate of the mass relative to some reference. Further, ΣF is the sum of all the forces acting on the body, normally divided into applied forces (gravity, springs, dampers, actuators, friction, wind resistance, rolling resistance...) and reactive forces (connections with other bodies and ground), see Fig. 2.1.

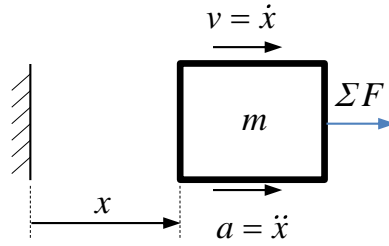


Figure 2.1 Translating body.

Beside gravity the most typical applied forces used in modeling of systems are spring force, damping force and friction force. The spring force, F_k , depends on the relative motion of the bodies that the spring is attached to, see Fig. 2.2.

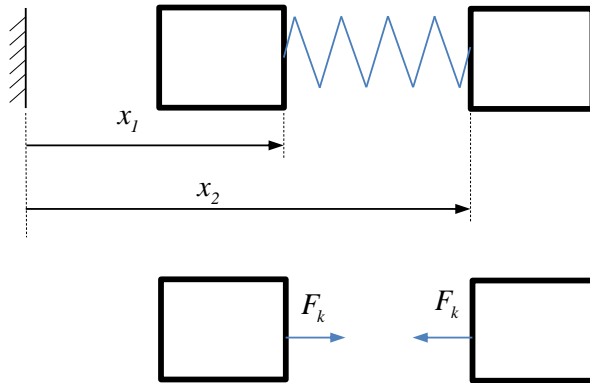


Figure 2.2 Spring force acting between two bodies.

If the spring is linear and can transmit both compression and tension then the expression for the spring force is:

$$F_k = k \cdot (x_2 - x_1 - L_0) \quad (2.2)$$

In (2.2), the undeformed length of the spring is denoted L_0 . Eq. (2.2) can be easily be extended to cover any type of spring force such as non-linear spring forces or spring forces that are only compression or tension. In (2.3), (2.4) and (2.5) three different spring force formulations are shown that take into account pure compression, pure tension as well as compression and tension with some slack.

$$F_k = \begin{cases} -k \cdot (x_1 + L_0 - x_2) & x_2 - x_1 \leq L_0 \\ 0 & x_2 - x_1 > L_0 \end{cases} \quad (2.3)$$

$$F_k = \begin{cases} 0 & x_2 - x_1 \leq L_0 \\ k \cdot (x_2 - x_1 - L_0) & x_2 - x_1 > L_0 \end{cases} \quad (2.4)$$

$$F_k = \begin{cases} -k \cdot (x_1 + L_1 - x_2) & L_1 > x_2 - x_1 \\ 0 & L_2 \geq x_2 - x_1 \geq L_1 \\ k \cdot (x_2 - x_1 - L_2) & x_2 - x_1 > L_2 \end{cases} \quad (2.5)$$

In (2.5) the total slack is $\Delta L = L_2 - L_1$. Non-linear springs are often used to model impact and may be formulated as a compressive force:

$$F_k = \begin{cases} -k \cdot (x_1 + L_0 - x_2)^n & x_2 - x_1 \leq L_0 \\ 0 & x_2 - x_1 > L_0 \end{cases} \quad (2.6)$$

where the exponent $n \neq 1$ introduces the non-linearity. Eqs. (2.3...6) are difficult to handle in analytical computations because they contain more than one expression. This is, however, no problem in time domain simulation because the different expressions are easily handled by means of if-then-else logic in the simulation model. Hence, the same spring force may be described by as many different expressions as needed, however, continuity across the intervals should be observed. Otherwise, the simulation may become unstable around the discontinuity.

Damping forces appear in any practical machine but its parameters are often more elusive than those of the spring force. Typically, it is introduced as illustrated in Fig. 2.3.

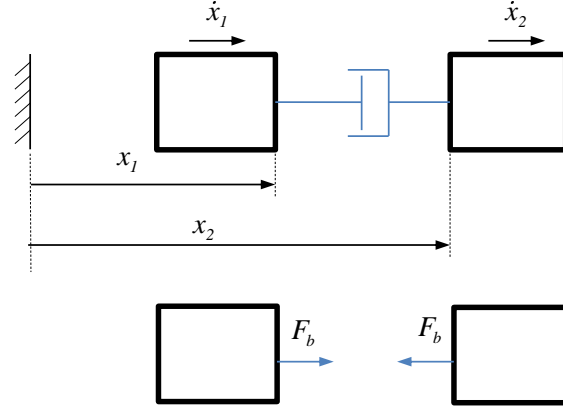


Figure 2.3 Damping force acting between two bodies.

A typical way of modeling damping is as a linear velocity dependent force, F_b :

$$F_b = b \cdot (\dot{x}_2 - \dot{x}_1) \quad (2.7)$$

In (2.7) the damping coefficient is denoted b . If the damping is modeled as part of impact phenomena then (2.7) is not suitable, because there will be a jump in damping force when the impact occurs. In that case a more suitable model also involves the overlap:

$$F_b = \begin{cases} b \cdot (\dot{x}_2 - \dot{x}_1) \cdot \sqrt{x_1 + L_0 - x_2} & x_2 - x_1 \leq L_0 \\ 0 & x_2 - x_1 > L_0 \end{cases} \quad (2.8)$$

Damping is viscous friction, however, a unified friction model that includes all effects does not exist. In fact, there is a wide range of different friction force models. What they do have in common is a dependency on the normal force, N and the relative velocity, $\dot{x}_2 - \dot{x}_1$, between the mating friction surfaces, see Fig. 2.4.

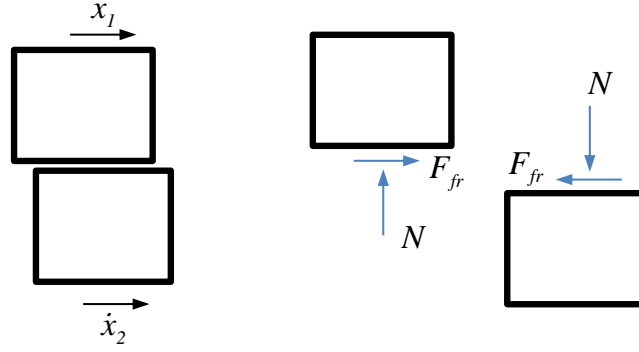


Figure 2.4 Normal and friction force acting between two bodies sliding on each other.

The ratio between the friction force and the normal force is defined as the friction coefficient:

$$F_{fr} = \mu \cdot N \quad (2.9)$$

One of the challenges with friction modeling is to ensure that the sign of the friction force, F_{fr} , changes with the sign of the relative velocity. A quite popular friction model that only uses one expression is based on the hyperbolic tangent function:

$$\mu = \mu_k \cdot \tanh\left(\frac{\dot{x}_2 - \dot{x}_1}{v_{mh}}\right) \quad (2.10)$$

In (2.10) μ_k is the kinematic friction coefficient and v_{mh} is a velocity parameter used to control the smoothness of the friction variation around zero relative velocity, see Fig. 2.5.

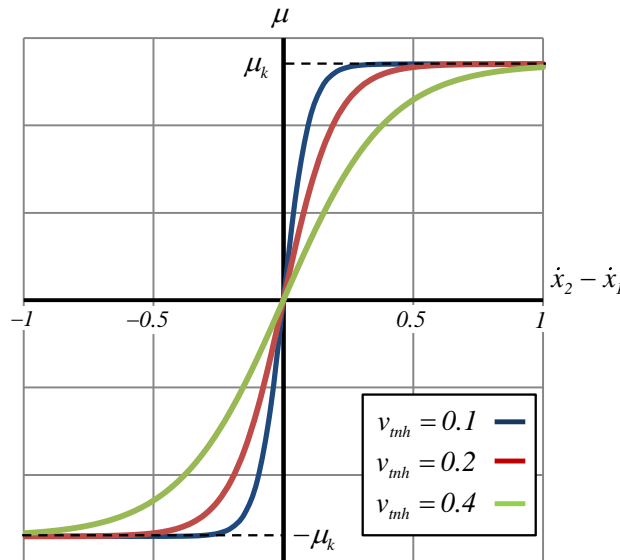


Figure 2.5 The influence of v_{mh} on the tanh friction model.

The main disadvantage with the tanh-friction model is that it does not return any friction force at zero relative velocity which is contrary to what is normally the case. Therefore, this friction model is most useful when modeling systems where the relative velocity oscillates around zero. Other models, such as Coulomb or Stribeck predicts friction at zero relative velocity. Coulomb friction states that:

$$\mu = \begin{cases} \mu_k & \dot{x}_2 - \dot{x}_1 \geq 0 \\ -\mu_k & \dot{x}_2 - \dot{x}_1 < 0 \end{cases} \quad (2.11)$$

whereas Stribeck friction takes into account that the friction coefficient quite often increases near zero relative velocity:

$$\mu = \begin{cases} (\mu_s - \mu_k) \cdot e^{-\left(\frac{\dot{x}_2 - \dot{x}_1}{v_{str}}\right)} + \mu_k & \dot{x}_2 - \dot{x}_1 \geq 0 \\ -(\mu_s - \mu_k) \cdot e^{-\left(\frac{\dot{x}_1 - \dot{x}_2}{v_{str}}\right)} - \mu_k & \dot{x}_2 - \dot{x}_1 < 0 \end{cases} \quad (2.12)$$

Here, μ_s is the friction coefficient at zero relative velocity and v_{str} is the Stribeck-velocity. Its influence on the variation of the friction coefficient is illustrated in Fig. 2.6.

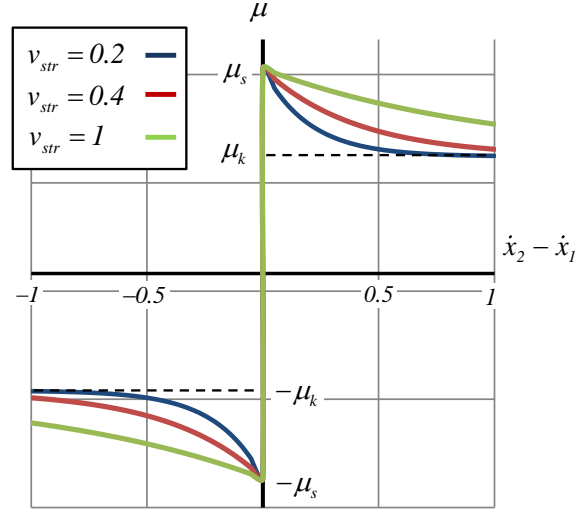


Figure 2.6 The influence of v_{str} on the Stribeck friction model.

Alternatively, the Stribeck friction model can also be formulated as:

$$\mu = \begin{cases} (\mu_s - \mu_k) \cdot e^{-\left(\frac{\dot{x}_2 - \dot{x}_1}{v_{str}}\right)^2} + \mu_k & \dot{x}_2 - \dot{x}_1 \geq 0 \\ -(\mu_s - \mu_k) \cdot e^{-\left(\frac{\dot{x}_1 - \dot{x}_2}{v_{str}}\right)^2} - \mu_k & \dot{x}_2 - \dot{x}_1 < 0 \end{cases} \quad (2.13)$$

The main disadvantage with Coulomb and Stribeck is that they introduce a discontinuity around zero relative velocity. This may easily cause the simulation to become unstable and therefore they are not very useful when modeling friction where the relative velocity passes through zero often. They can be employed for models where the velocity approaches zero from either negative or positive direction but maintains the same sign.

If the simulation must handle both longer periods with zero relative velocity as well as frequent sign changes in the relative velocity then a standard LuGre friction model can be used. In the LuGre friction model an extra state is introduced which describes the deflection between the friction surfaces at very

low relative velocity. This extra deflection is normally referred to as z and a standard Luge formulation is:

$$\mu = \sigma_0 \cdot z + \sigma_1 \cdot \dot{z} \quad (2.14)$$

$$\dot{z} = v_{rel} - \sigma_0 \cdot \frac{|v_{rel}|}{g(v_{rel})} \cdot z \quad (2.15)$$

$$g(v_{rel}) = (\mu_s - \mu_k) \cdot e^{-\left(\frac{v_{rel}}{v_{str}}\right)^2} + \mu_k \quad (2.16)$$

$$v_{rel} = \dot{x}_2 - \dot{x}_1 \quad (2.17)$$

The main disadvantage with Luge as compared to the earlier models is simply the increased complexity with an extra state variable and more parameters.

The governing dynamic equations for body that is rotating around a fixed point is:

$$J_o \cdot \ddot{\theta} = \Sigma M_o \quad (2.18)$$

In (2.18) J_o is the mass moment of inertia with respect to the fixed point of rotation, see Fig. 2.7. Furthermore, θ is the rotation of the body measured relative to some reference coordinate and ΣM_o is the sum of the force moment of all the forces acting on the body with respect to the fixed point of rotation and all the moments (force couples) acting on the body.

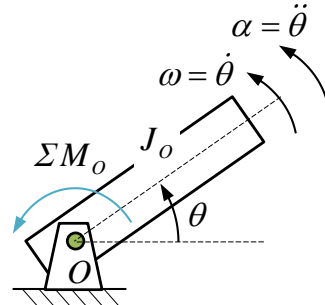


Figure 2.7 Rotating body.

For a rotating body the same type of equations regarding spring forces, damping forces and friction forces can be set up, simply replacing x and \dot{x} with θ and $\dot{\theta}$, and by changing forces into torques. As an example, (2.2) and (2.7) can be reformulated as equations that yield a spring torque and a damping torque, respectively, between two rotating bodies. This is formulated below:

$$M_k = k_\theta \cdot (\theta_2 - \theta_1 - \beta_0) \quad (2.19)$$

$$M_b = b_\theta \cdot (\dot{\theta}_2 - \dot{\theta}_1) \quad (2.20)$$

In (2.19) and (2.20) the torsional stiffness, k_θ , and torsional, b_θ , damping are introduced, see also Fig. 2.8.

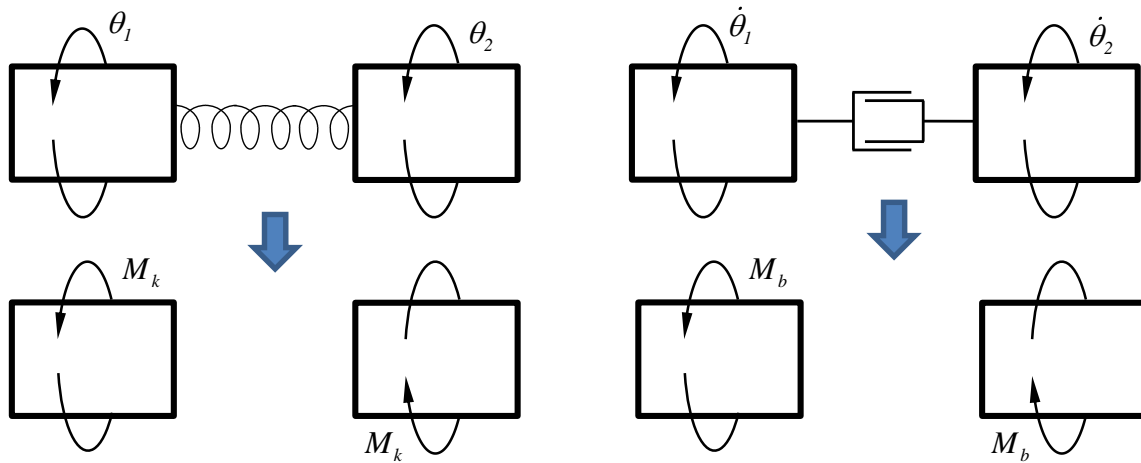


Figure 2.8 Spring and damping torque acting between two bodies.

Friction is a little more complicated since it requires the size of the reactive force in fixed point of rotation and extra information regarding the pin radius. A simple model might look as follows:

$$M_{fr} = M_{\mu} \cdot \tanh\left(\frac{\dot{\theta}_2 - \dot{\theta}_1}{\omega_{mh}}\right) \quad (2.21)$$

where M_{μ} is the maximum friction moment and ω_{mh} shapes the friction-velocity curve in the same way as v_{mh} in (2.10).

CHAPTER 3: Two-Dimensional Mechanical Systems

The governing dynamic equations for a two-dimensional (planar) body are:

$$\begin{aligned} m \cdot \ddot{\vec{r}}_G &= \Sigma \vec{F} \\ J_G \cdot \ddot{\theta} &= \Sigma M_G \end{aligned} \quad (3.1)$$

In (3.1) m is the mass and J_G is the mass moment of inertia with respect to the mass center of the body. Furthermore, \vec{r}_G is the coordinates of the mass center of the body and θ is the rotation of the body, both measured relative to some reference coordinate system. $\Sigma \vec{F}$ is the sum of all the forces acting on the body and ΣM_G is the sum of all moments acting on the body.

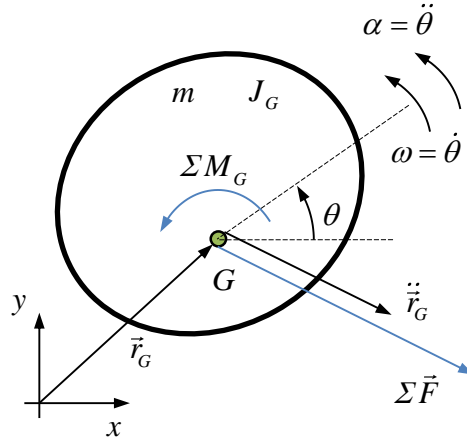


Figure 3.1 Planar body.

Often the rotational degree of freedom can be ignored if the body is subjected to pure translation or if the rotation is unimportant. In that case the dynamic equations reduce to:

$$m \cdot \ddot{\vec{r}}_G = \Sigma \vec{F} \quad (3.2)$$

When modeling forces in 2-dimensions it is important to recall that any force potentially consists of two components (typically an x- and a y-component). Often one of these components will be zero, but any modeling effort should include an assessment of both components.

A simple linear spring attached to two bodies, see Fig. 3.2 to the left, can be modeled as follows:

$$\vec{F}_k = k \cdot (L - L_0) \cdot \vec{e}_{A \rightarrow B} \quad (3.3)$$

Where L_0 is the undeformed length of the spring and $\vec{e}_{A \rightarrow B}$ is the unit vector pointing from A to B:

$$L = |\vec{r}_B - \vec{r}_A| \quad (3.4)$$

$$\vec{e}_{A \rightarrow B} = \frac{\vec{r}_B - \vec{r}_A}{L} \quad (3.5)$$

Similarly, a linear damper, see Fig. 3.2 to the right, may be modeled in the following way

$$\vec{F}_b = b \cdot v_{A \rightarrow B} \cdot \vec{e}_{A \rightarrow B} \quad (3.6)$$

Where L_0 is the undeformed length of the spring and

$$v_{A \rightarrow B} = (\dot{\vec{r}}_B - \dot{\vec{r}}_A) \cdot \vec{e}_{A \rightarrow B} \quad (3.7)$$

The velocity $v_{A \rightarrow B}$ is simply the velocity difference of the points A and B projected onto the direction of action of the damper.

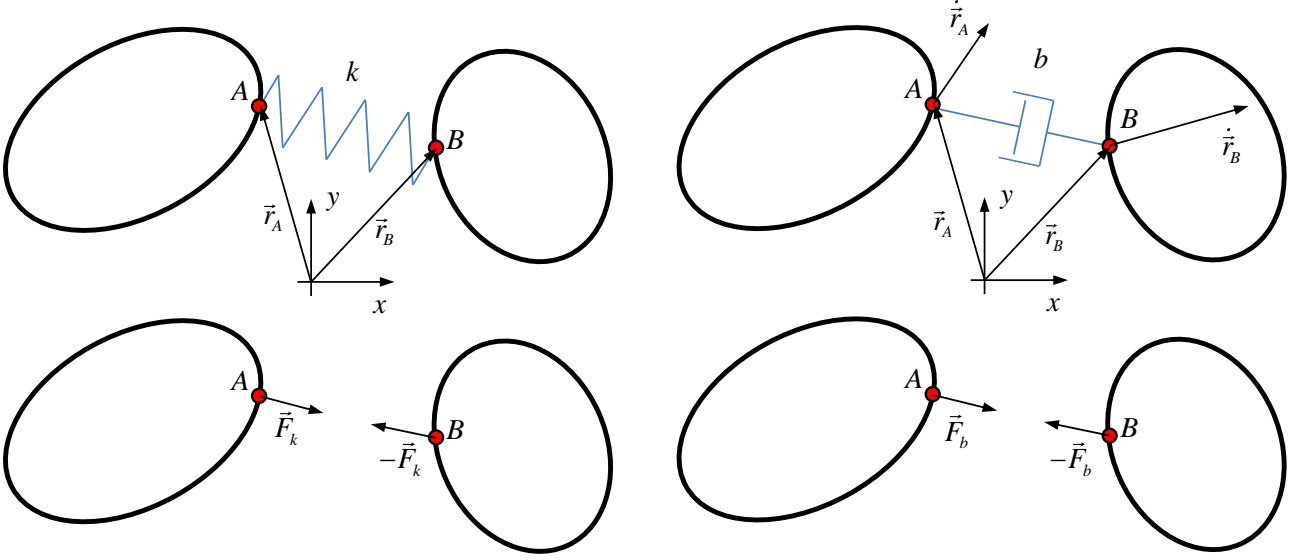


Figure 3.2 Spring and damping force in 2-dimensions.

A special type of friction is the one related to tire-ground interaction. It is normally referred to as traction and is used to model vehicles. It requires that the model contains both a rotational and a translational degree of freedom of the wheel of the tire, see Fig. 3.3. The friction force can be modeled according to (2.9), however, the friction coefficient is not a function of the absolute speed but of a dimensionless variable normally referred to as the slip:

$$\mu = \mu_k \cdot \tanh\left(\frac{s}{s_0}\right) \quad (3.8)$$

$$s = \frac{r \cdot \omega - v}{v_0} \quad (3.9)$$

$$v_0 = \max \begin{cases} |v| \\ |r \cdot \omega| \\ v_\varepsilon \end{cases} \quad (3.10)$$

When the wheel is rolling, $v = -r \cdot \omega$ (notice that the angular velocity is defined as positive when the wheel is rotating clockwise), the slip will be zero and there is no traction. However, as soon as this equality no longer holds the tire-ground connection will produce either an accelerating or a braking traction force depending on the sign of the slip.

In (3.8) the parameter s_0 is a parameter that depends on the tire-ground combination. It will typically have values within the interval $0.02 \leq s_0 \leq 0.2$ with low values for rigid tires on concrete or asphalt and high values for flexible tires on soil or grass. In (3.10) the parameter v_ϵ is simply a small value that ensures that the reference velocity, v_0 , is never zero thereby avoiding a singularity in (3.9) when the tire neither translates nor rotates. This gives the basic disadvantage that there can be no traction force when the tire is at a standstill but it is very convenient to model the acceleration and braking of vehicles.

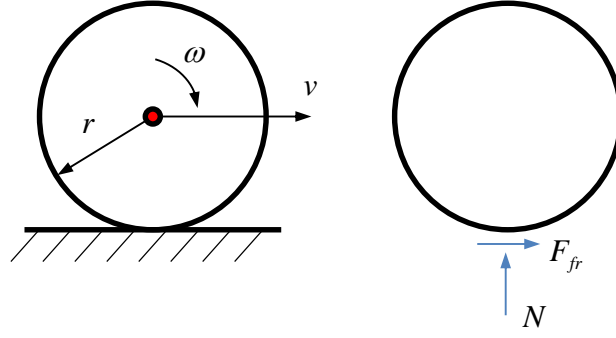


Figure 3.3 Normal and traction force acting on wheel.

In some cases, one or more of the bodies of a system will undergo deflections that are so large that they should not be ignored. In general, combining the small motion related to the deformability of bodies with their gross = rigid body motion is a both theoretically complicated and numerically difficult task. In this context, a simple lumped approach to the most common flexible bodies. In general, the approach is based on a subdivision of the flexible element to be investigated into a number of rigid body segments. For each rigid body segment the usual dynamic equilibrium equations, i.e. (3.1), may be applied and the flexibility is introduced by adding springs between adjacent bodies, whose stiffness reflects the overall stiffness of the body.

A beam subjected to bending may be modelled by dividing it into a number of rigid body segments with a revolute joint as well as a bending spring at each end, see Fig. 3.4.

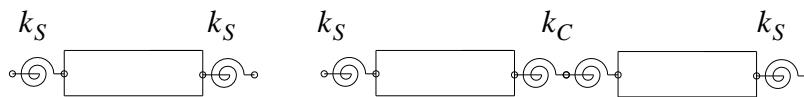


Figure 3.4 To the left is shown a single beam segment with its bending springs. To the right, 2 joined segments are shown.

The rotational spring stiffness of a beam segment is given as:

$$k_S = \frac{2 \cdot E \cdot I}{L} \quad (3.11)$$

In (3.11), E is Youngs modulus, I is the area moment of inertia and L is the length of the segment. The number of segments a beam is divided into should be large enough to cover the eigenfrequencies properly and at the same time be as limited as possible to reduce computation time. For two adjoining beam segments the resulting connecting spring stiffness, see Fig. 3.4, becomes:

$$k_C = \frac{I}{\frac{I}{k_{S,1}} + \frac{I}{k_{S,2}}} \quad (3.12)$$

It is the effective spring stiffness that will apply moments to the two adjoining rigid body segments proportional to their relative rotation. A rod subjected to axial loading may similarly be divided into rigid body segments, with axial springs at both ends. The spring stiffness for the axial springs are:

$$k_S = \frac{2 \cdot E \cdot A}{L} \quad (3.13)$$

In (3.13), E is Young's modulus, A is the cross-sectional area and L is the length of the segment. The resulting axial spring stiffness of 2 adjoining elements may be determined using (3.12). A beam with circular cross section geometry and loaded in torsion may similarly be divided into rigid body segments with torsional springs at both ends. The spring stiffness for the torsional springs are:

$$k_S = \frac{2 \cdot G \cdot I_p}{L} \quad (3.14)$$

In (3.14), G is the shear modulus, I_p is the polar moment of inertia and L is the length of the segment. As for the other 2 types of segments, the resulting torsional spring stiffness of 2 adjoining elements may be determined using (3.12).

CHAPTER 4: Effective Inertia and Effective Load

In Chapter 2 and 3 mechanical systems have been investigated. Seen from a modeling point of view, the mechanical system can often become very complex as compared to the pneumatic/hydraulic/electrical actuation and control system. It is, however, very often possible to reduce the complexity of the mechanical system substantially. Actually, for most systems it is possible to compute an effective inertia and an effective load for each individual actuator. This greatly reduces the complexity of the models and, simultaneously, gives a better overview of the system. In this context the actuators are either rotating (a motor) or translating (a cylinder/linear actuator). The actuators may be pneumatic, hydraulic or electrical, it does not matter, the effective inertia and effective load is applicable to any actuating device trying to manipulate a machine.

For the motor a simplified dynamic equation can be set up:

$$J_{eff} \cdot \ddot{\theta} = M_M - M_{eff} \quad (4.1)$$

In (4.1) the sign conventions are as follows: rotation of the shaft, θ (and its time derivatives; $\dot{\theta}$ and $\ddot{\theta}$), is defined as positive in the same direction as the motor torque, τ_M . The motor torque is generated hydraulically or electromagnetically via interactions between a stator and a rotor that is rigidly connected to the output shaft of the motor. The applied moment (the load) on the output shaft, M_{eff} , is positive in the opposite direction.

The effective mass moment of inertia, J_{eff} , and the applied moment, M , must be related to the output shaft of the motor. In general, they are both functions of θ and $\dot{\theta}$, and may be determined from energy considerations. The applied moment may be computed as follows:

$$M_{eff} = \frac{-dW_{ext}}{d\theta} \quad (4.2)$$

In (4.2) W_{ext} is the work done by the external forces/moments (gravity, friction, rolling resistance etc.) on the mechanical system actuated by the motor and dW_{ext} is the work done for a small rotation of the motor, $d\theta$.

The effective mass moment of inertia may be computed according to:

$$J_{eff} = \frac{2 \cdot E_{kin}}{\dot{\theta}^2} \quad (4.3)$$

In (4.3) E_{kin} is the kinetic energy of the mechanical system that should be formulated as a function of the geometrical data and the angular velocity of the motor, $\dot{\theta}$. Hence, the target is to identify all inertias that are rigidly connected to the output shaft of the motor and formulate their total kinetic energy as: $E_{kin} = E_{kin}(\dot{\theta}^2)$. As an example, consider the system shown in Fig. 4.1.

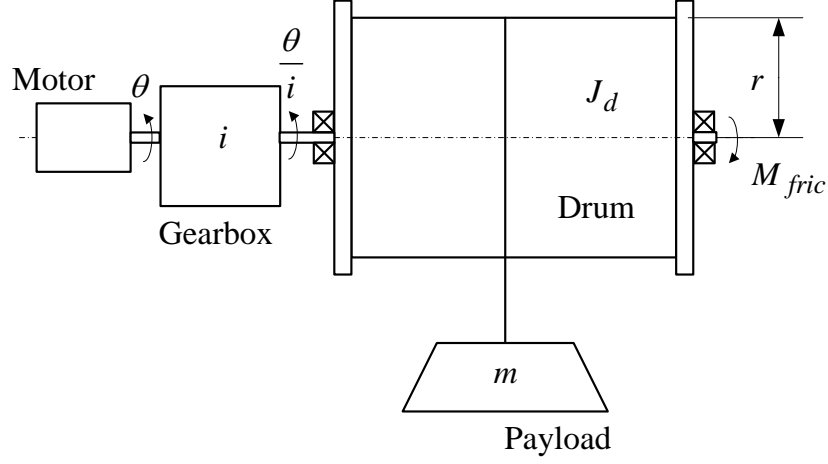


Figure 4.1 A motor is driving a winchdrum via a gearbox. The drum is connected to a payload and is also subjected to a certain rotational friction.

For an infinitesimal rotation of the motor shaft the work done on the system by the external loading can be computed and, subsequently, the moment applied to the output shaft of the motor:

$$\begin{aligned}
 W_{ext} &= -m \cdot g \cdot dy - M_{fric} \cdot d\theta_d = -m \cdot g \cdot r \cdot \frac{d\theta}{i} - M_{fric} \cdot \frac{d\theta}{i} \\
 \Downarrow \\
 M_{eff} &= \frac{m \cdot g \cdot r + M_{fric}}{i}
 \end{aligned} \tag{4.4}$$

The effective mass moment of inertia may be computed as:

$$\begin{aligned}
 E_{kin} &= \frac{1}{2} \cdot J_{rot} \cdot \dot{\theta}^2 + \frac{1}{2} \cdot J_d \cdot \dot{\theta}_d^2 + \frac{1}{2} \cdot m \cdot v^2 = \frac{1}{2} \cdot J_{rot} \cdot \dot{\theta}^2 + \frac{1}{2} \cdot J_d \cdot \left(\frac{\dot{\theta}}{i} \right)^2 + \frac{1}{2} \cdot m \cdot \left(r \cdot \frac{\dot{\theta}}{i} \right)^2 \\
 \Downarrow \\
 J_{eff} &= J_{rot} + \frac{J_d}{i^2} + \frac{m \cdot r^2}{i^2}
 \end{aligned} \tag{4.5}$$

Notice, that the mass moments of inertia of the gearbox components have been neglected. This is normally a reasonable assumption although for high speed robotic applications it is customary to include the inertia of the gearbox.

Next, consider the system shown in Fig. 4.2. It is a four-wheel drive vehicle subjected to a total rolling resistance of F_R , and propelled by a hydraulic motor via a geared chain drive $i = \frac{z_2}{z_1}$. Due to symmetry, the vehicle is modeled as a planar system.

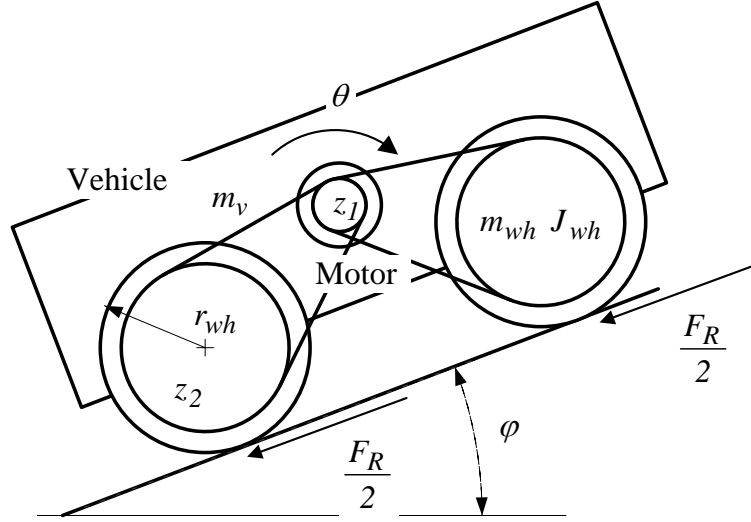


Figure 4.2 A four-wheel drive vehicle propelled up an incline by a traction motor is shown. Power is transmitted from the motor to the wheels via chain drives.

As in the previous example a infinitesimal rotation of the motor is used to determine the moment applied to the output shaft of the motor:

$$\begin{aligned}
 W_{ext} &= -(m_v + 2 \cdot m_{wh}) \cdot g \cdot dy - F_R \cdot ds = -(m_v + 2 \cdot m_{wh}) \cdot g \cdot r_{wh} \cdot \frac{d\theta}{i} \cdot \sin \phi - F_R \cdot r_{wh} \cdot \frac{d\theta}{i} \\
 \Downarrow \\
 M_{eff} &= \frac{r_{wh} \cdot \{F_R + (m_v + 2 \cdot m_{wh}) \cdot g \cdot \sin \phi\}}{i}
 \end{aligned} \tag{4.6}$$

The effective mass moment of inertia (ignoring the inertia of the motor rotor):

$$\begin{aligned}
 E_{kin} &= 2 \cdot \frac{1}{2} \cdot J_{wh} \cdot \dot{\theta}_{wh}^2 + \frac{1}{2} \cdot (2 \cdot m_{wh} + m_v) \cdot v^2 = J_{wh} \cdot \left(\frac{\dot{\theta}}{i}\right)^2 + \frac{1}{2} \cdot (2 \cdot m_{wh} + m_v) \cdot \left(r_{wh} \cdot \frac{\dot{\theta}}{i}\right)^2 \\
 \Downarrow \\
 J_{eff} &= \frac{2 \cdot J_{wh}}{i^2} + \frac{(2 \cdot m_{wh} + m_v) \cdot r_{wh}^2}{i^2}
 \end{aligned} \tag{4.7}$$

Having determined the effective mass moment of inertia and the effective external load the computation of the angular acceleration (as is needed in time domain simulation) becomes very simple:

$$\ddot{\theta} = \frac{M_M - M_{eff}}{J_{eff}} \tag{4.8}$$

Under normal conditions the torque provided by the motor can be computed from the state variables, $M_M = M_M(\theta, \dot{\theta})$. In fact, (4.8) is always applicable as long as the three parameters on the right hand side can be computed from the state variables which is usually the case.

Similar to that of the motor a simplified dynamic equation can be set up for the linear actuator:

$$m_{eff} \cdot \ddot{x} = F_L - F_{eff} \quad (4.9)$$

In (4.9) the sign conventions are as follows: linear travel, x (and its time derivatives; \dot{x} and \ddot{x}), is defined as positive when the linear actuator is extracting. The force delivered by the linear actuator, F_L , is positive in the same direction as, x , and the applied force on the piston of the actuator, F_{eff} , is positive in the opposite direction.

The effective mass, m_{eff} , and the effective applied force, F_{eff} , must be related to the piston of the linear actuator. In general, they are both functions of x and \dot{x} , and may, similar to the values associated with the motor, be determined from energy considerations. The effective applied force may be computed as follows:

$$F_{eff} = \frac{-dW_{ext}}{dx} \quad (4.10)$$

The effective mass may be computed according to:

$$m_{eff} = \frac{2 \cdot E_{kin}}{\dot{x}^2} \quad (4.11)$$

As an example, consider the system shown in Fig. 4.3.

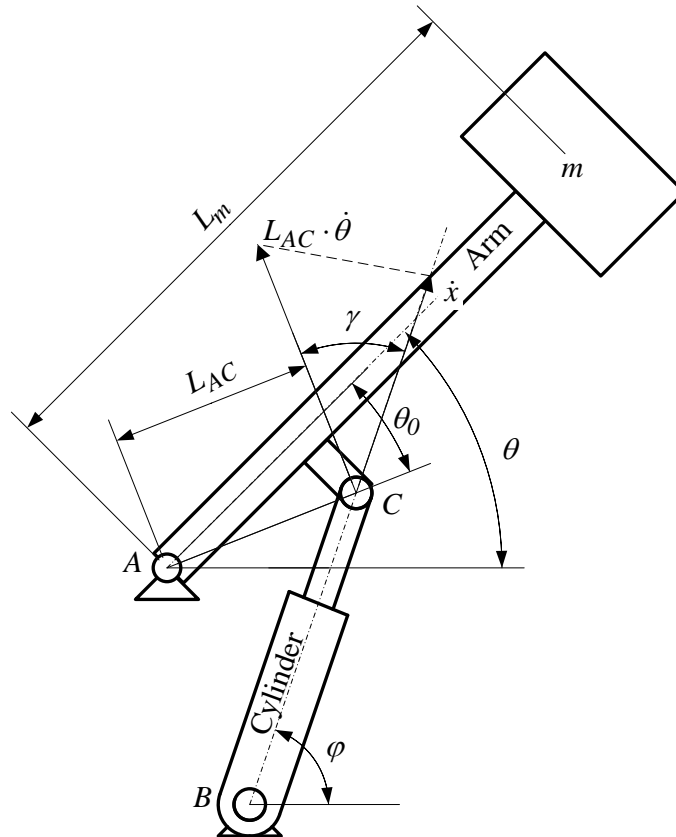


Figure 4.3 A linear actuator is rotating an arm with a payload at the end.

For a infinitesimal extraction of the cylinder the work done on the system by the external loading can be computed and, subsequently, the force applied to the actuator piston:

$$\begin{aligned}
 W_{ext} &= -m \cdot g \cdot dy_m = -m \cdot g \cdot L_m \cdot \cos \theta \cdot d\theta \\
 \Downarrow \\
 F_{eff} &= m \cdot g \cdot L_m \cdot \cos \theta \cdot \frac{\dot{\theta}}{\dot{x}} = m \cdot g \cdot L_m \cdot \cos \theta \cdot \frac{\dot{\theta}}{L_{AC} \cdot \dot{\theta} \cdot \cos \gamma} = m \cdot g \cdot \frac{L_m \cdot \cos \theta}{L_{AC} \cdot \cos \gamma} \\
 \gamma &= \theta - \theta_0 + \frac{\pi}{2} - \phi \quad \phi = \tan^{-1} \left(\frac{y_C - y_B}{x_C - x_B} \right)
 \end{aligned} \tag{4.12}$$

In (4.12) it is utilized that the arm and the piston must have the same absolute velocity in point C. The effective mass may be computed as:

$$\begin{aligned}
 E_{kin} &= \frac{1}{2} \cdot m \cdot v^2 = \frac{1}{2} \cdot m \cdot (L_m \cdot \dot{\theta})^2 = \frac{1}{2} \cdot m \cdot L_m^2 \cdot \left(\frac{\dot{x}}{L_{AC} \cdot \cos \gamma} \right)^2 \\
 \Downarrow \\
 m_{eff} &= m \cdot \left(\frac{L_m}{L_{AC} \cdot \cos \gamma} \right)^2
 \end{aligned} \tag{4.13}$$

Again, the computation of the effective mass and effective applied force yields a very simple equation for the computation of the acceleration in the time domain simulation:

$$\ddot{x} = \frac{F_L - F_{eff}}{m_{eff}} \tag{4.14}$$

Clearly, the effective applied force is a function of the position, $F_{eff} = F_{eff}(x)$ and the effective mass is a function of the position and velocity, $m_{eff} = m_{eff}(x, \dot{x})$, but this is no problem in time domain simulation since these parameters correspond to the state variables. As for the motor, it will also normally be possible to derive the actuator force, F_L , from the state variables.

CHAPTER 5: Actuation Systems

5.1 Hydraulic actuation

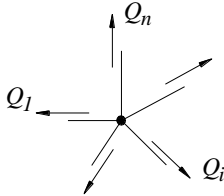
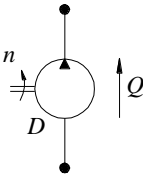
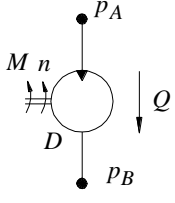
In this chapter, we will discuss modeling of hydraulic and electrical actuation systems. We start with hydraulics. Physically, dynamic simulation of a hydraulic system is mainly characterized by:

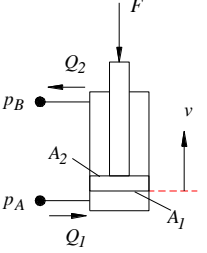
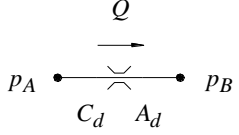
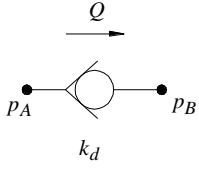
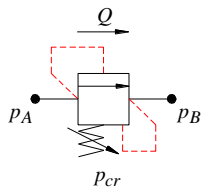
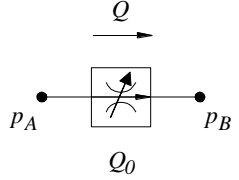
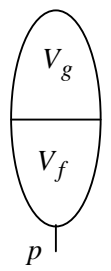
- acceleration of the mechanical system
- compressible fluid
- dynamics of valves

Dynamic simulation allows for a much more detailed investigation of the performance of the hydraulic system, however, it does also require that further estimations of especially system damping and oil stiffness in order to predict behavior correctly. In practical simulation, it is important not to include all possible dynamics. As an example, if some part of the mechanical system has negligible inertia (mass or mass moment of inertia) or is moving very slowly, then static equilibrium is adequate. Similarly, if we have very small volumes of fluid we should, normally, avoid modeling them as compressible since this will increase computational time without adding new information to the simulation. Finally, many valves can be considered infinitely fast as compared to the remaining hydraulic-mechanical system, and in that case, they should be modeled using steady state equations. Steady state equations are algebraic equations whereas dynamic equations are differential equations. Hence, in practice we often do time domain simulation by solving a mixed set of differentials and algebraic equations. The differential equations are time dependent, i.e., the problem is an initial value problem, where the pressure in the pressure nodes, the volumes of the accumulators and the position and velocity of all mechanical degrees of freedom must be known at the start of simulation.

In Tab. 5.1 the steady state equations are listed for the most common hydraulic components.

Table 5.1 Steady state equations for basic components

Description	Symbol	Equations SI-units	Equations Hyd-units
Pressure node		$\sum_{i=1}^n Q_i = 0$	$\sum_{i=1}^n Q_i = 0$
Pump		$Q = n \cdot D$	$Q = \frac{n \cdot D}{1000}$
Motor		$Q = n \cdot D$ $M = \frac{D \cdot (p_A - p_B)}{2 \cdot \pi}$	$Q = \frac{n \cdot D}{1000}$ $M = 0.0159 \cdot D \cdot (p_A - p_B)$
Cylinder		$Q_l = v \cdot A_l$	$Q_l = 0.06 \cdot v \cdot A_l$

		$Q_2 = v \cdot A_2$ $F = p_A \cdot A_1 - p_B \cdot A_2$	$Q_2 = 0.06 \cdot v \cdot A_2$ $F = \frac{p_A \cdot A_1 - p_B \cdot A_2}{10}$
Orifice		$Q = C_d \cdot A_d \cdot \sqrt{\frac{2}{\rho} \cdot (p_A - p_B)}$	$Q = 0.89 \cdot C_d \cdot A_d \cdot \sqrt{p_A - p_B}$
Check valve		$Q = 0 \quad p_A \leq p_B$ $Q > 0 \quad p_A = p_B$	$Q = 0 \quad p_A \leq p_B$ $Q > 0 \quad p_A = p_B$
Pressure relief valve		$Q = 0 \quad p_A - p_B \leq p_{cr}$ $Q > 0 \quad p_A - p_B = p_{cr}$	$Q = 0 \quad p_A - p_B \leq p_{cr}$ $Q > 0 \quad p_A - p_B = p_{cr}$
2-way flow control valve		$Q = \frac{Q_0}{\sqrt{\Delta p_0}} \cdot \sqrt{p_A - p_B} \quad p_A - p_B \leq \Delta p_0$ $Q = Q_0 \quad p_A - p_B > \Delta p_0$	$Q = \frac{Q_0}{\sqrt{\Delta p_0}} \cdot \sqrt{p_A - p_B} \quad p_A - p_B \leq \Delta p_0$ $Q = Q_0 \quad p_A - p_B > \Delta p_0$
Accumulator		$p \cdot V_g^n = p_0 \cdot V_0^n$ $V_a = V_f + V_g = cst$	$p \cdot V_g^n = p_0 \cdot V_0^n$ $V_a = V_f + V_g = cst$

In Table 5.1 the so-called orifice equation is used frequently. Orifices are small openings that the oil is throttled across, see Fig. 5.1.

The equation linking the flow through an orifice to the subsequent pressure drop in the flow is:

$$Q = C_d \cdot A_d \cdot \sqrt{\frac{2}{\rho} \cdot (p_1 - p_2)} \quad (5.1)$$

In (5.1) Q is the volume flow, C_d is the discharge coefficient. It is a geometry dependent dimensionless constant that indicates how effectively the flow passes the nominal discharge area, A_d . Furthermore, ρ is the mass density of the oil, and $p_1 - p_2$ is the pressure drop across the orifice. Hence, flow will either be entering or leaving a volume across an orifice depending on whether the pressure within the volume is larger than that on the other side or vice versa.

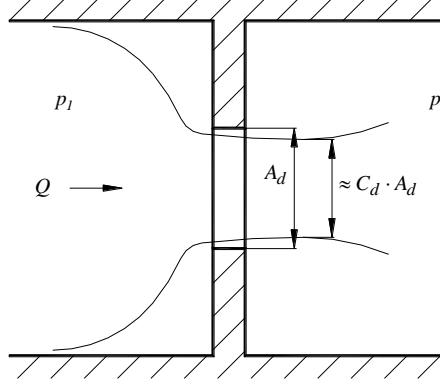


Figure 5.1 Orifice.

It is mostly the volume flow and the pressure drop that varies for an orifice. Therefore, (5.1) is often be employed in a simpler and more practical formulation:

$$Q = K_v \cdot \text{SIGN}(\Delta p) \cdot \sqrt{|\Delta p|} \quad (5.2)$$

Where $\text{SIGN}(\Delta p) = \pm 1$ is the sign of the pressure drop and an orifice constant

$$K_v = C_d \cdot A_d \cdot \sqrt{\frac{2}{\rho}} \quad (5.3)$$

Using (5.2) ensures that the volume flow can run in both directions through the orifice and it also avoids taking the square-root of a negative number. Often, the data for a certain orifice is only given as a simultaneous set of volume flow and pressure drop: $(Q_0, \Delta p_0)$. Based on that information the orifice can be determined:

$$K_v = \frac{Q_0}{\sqrt{\Delta p_0}} \quad (5.4)$$

In many cases, the orifice is variable, i.e., its opening is defined by the motion of one or more spools within a valve body. In that case it is customary to model the valve constant as linear with a dimensionless orifice opening, u :

$$Q = K_v \cdot u \cdot \text{SIGN}(\Delta p) \cdot \sqrt{|\Delta p|} \quad (5.5)$$

Hence, the orifice is closed when $u = 0$ and fully open when $u = 1$. Again, the orifice constant can be determined from (5.4) based on a simultaneous set of volume flow and pressure drop $(Q_0, \Delta p_0)$ - **but it must be for the fully opened orifice.**

Most notable is the difference when considering flow continuity of a pressure node. If the fluid is compressible then conservation of mass yields the following differential equation for a volume of fluid = pressure node:

$$\dot{p} = \frac{\beta \cdot (Q - \dot{V})}{V} \quad (5.6)$$

This equation gives the pressure gradient in a given volume of fluid, V . The net-flow into the volume is Q (positive if flow enters the volume) and the time derivative of the expansion (displacement flow) is \dot{V} (positive if the volume is expanding). If this value is positive at a certain time the pressure is going up and vice versa. The effective stiffness of the fluid, which greatly depends on temperature, dissolved air, hosing and tubing, is β . In this case we have a 1st-order differential equation and only one initial condition is required, namely the initial pressure in the volume: p .

As an example consider the volume shown in Fig. 5.2 which is bounded by two cylinders, two motors, two pumps and two orifices. Normally, a single volume is not bounded by so many components but here it is done for demonstrational purposes.

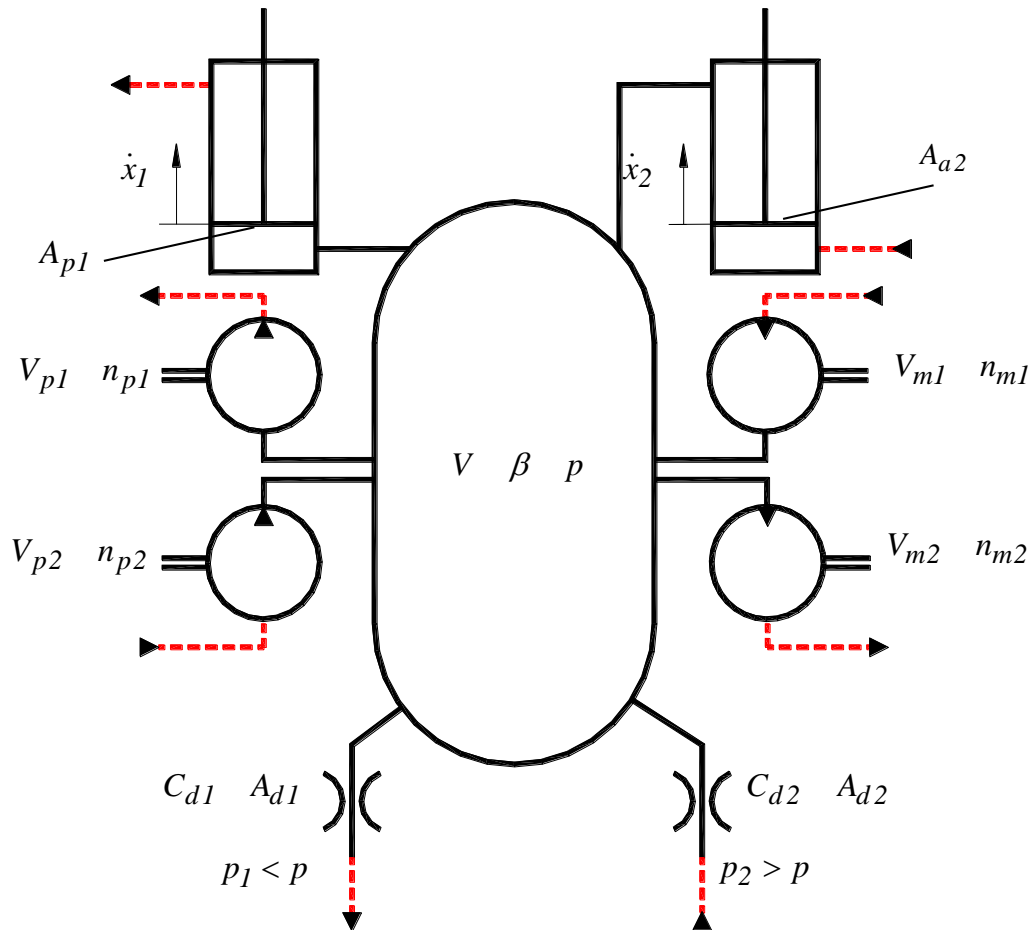


Figure 5.2 A volume bounded by cylinders, motors, pumps and orifices.

The differential equations for the pressure gradient of this volume may be written using (5.6) with the following values for Q and \dot{V} :

$$Q = -D_{p1} \cdot n_{p1} + D_{p2} \cdot n_{p2} + D_{m1} \cdot n_{m1} - D_{m2} \cdot n_{m2} - C_{d1} \cdot A_{d1} \cdot \sqrt{\frac{2}{\rho} \cdot (p - p_1)} + C_{d2} \cdot A_{d2} \cdot \sqrt{\frac{2}{\rho} \cdot (p_2 - p)} \quad (5.7)$$

$$\dot{V} = A_{p1} \cdot \dot{x}_1 - A_{a2} \cdot \dot{x}_2 \quad (5.8)$$

Displacement flows in volumes are typically caused by hydraulic cylinders or accumulators. If we consider the steady state equations for an accumulator then the time derivative yields a linear equation containing both the pressure and volume gradient:

$$\begin{aligned} \dot{p} \cdot V_g^n + n \cdot p \cdot V_g^{n-1} \cdot \dot{V}_g &= 0 \quad \& \quad \dot{V}_g + \dot{V}_f = 0 \\ \Downarrow \\ \dot{V}_f &= \frac{\dot{p}}{n \cdot p} \cdot V_g \end{aligned} \quad (5.9)$$

In the above equation, the polytropic exponent n depends on whether the compression/decompression of the accumulator is predominantly adiabatic or isothermal. In most time domain simulation, it is safe to assume adiabatic conditions in which case the exponent is approximately constant $n = 1.3$.

In (5.9) we introduce another state variable, namely the fluid volume of the accumulator. The initial fluid volume is readily derived from the total accumulator volume (which is constant), the preload conditions and the current pressure:

$$V_f = V_a - \left(\frac{p_0}{p} \right)^{\frac{1}{n}} \cdot V_0 \quad V_g = \left(\frac{p_0}{p} \right)^{\frac{1}{n}} \cdot V_0 \quad (5.10)$$

Hence, if a volume is connected to an accumulator it has two states: the pressure and the fluid volume of the accumulator and their gradients must be solved simultaneously using (5.9) and (5.10).

In general, a purely hydraulic system may be solved numerically in the following steps:

1. Identify all volumes in the system and set up the pressure build up equation for each and identify all accumulators and set up the (Circuit diagrams useful here).
2. Identify all orifices and their dependency on the motion of mechanical parts in valves.
3. Determine initial pressure for each volume and initial fluid volume for each accumulator.
4. Calculate pressure gradients for each volume and volume gradient for each accumulator.
5. Calculate acceleration of all movable mechanical parts in valves.
6. Update pressure in each volume and volume of each accumulator.
7. Update position and velocity of all movable mechanical parts in valves.
8. If the analysis is not yet concluded then go back to 4.

Typically some mechanical bodies, cylinders or valves will be part of the system. This means that their position and velocity must be updated simultaneously in order to update volumes, orifice flows and net-flows into volumes for the next step.

As may readily be observed from (5.6..9) the position and velocity of the mechanical system is needed in order to do a dynamic simulation of the hydraulic system. In fact, any type of dynamic simulation of a hydraulic system requires a simultaneous dynamic simulation of the actuated mechanical system

and, depending on the level of detail, also of the movable mechanical parts within the hydraulic valves. Hence, dynamic simulation of hydraulics is closely connected to dynamic simulation of mechanics.

As an example let us consider the hydraulic system shown in Fig. 5.3 and set up the combined set of differential and algebraic equations.

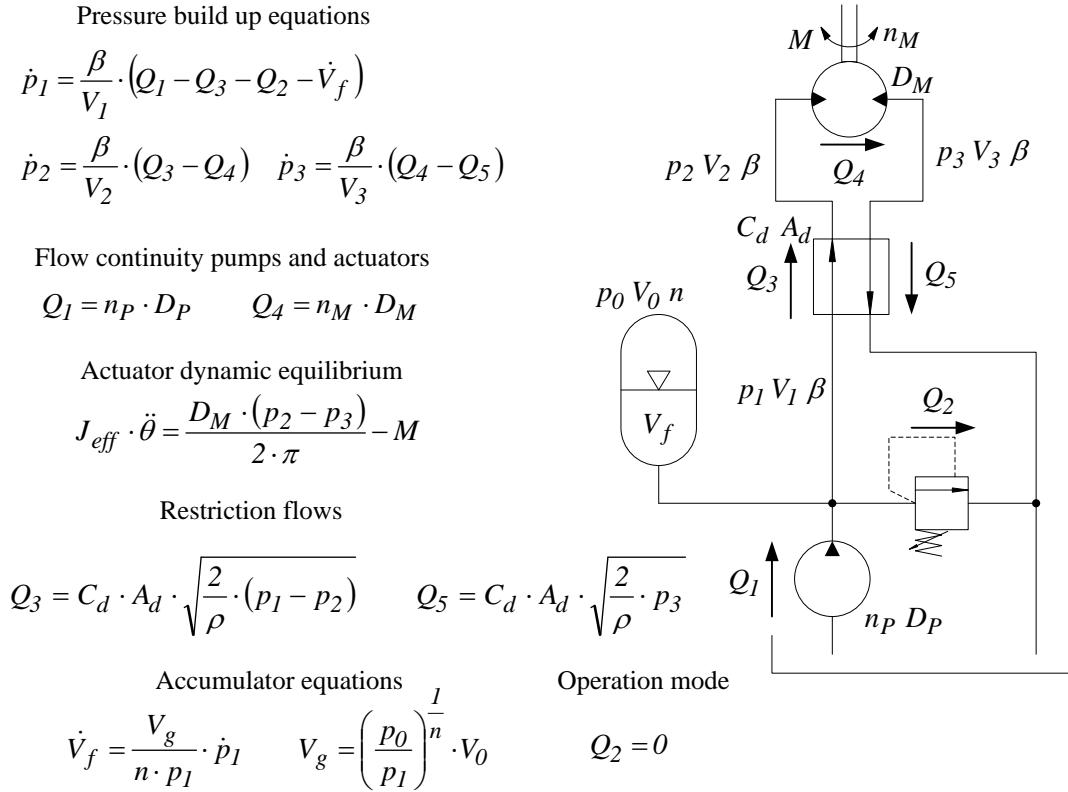


Figure 5.3 Hydraulic circuit and corresponding set of differential and algebraic equations.

In order to run a simulation obviously some input must be prescribed in time. Typically, this can be the load on the motor output shaft, the motion of the directional control valve and the angular speed of the pump.

5.2 Electrical actuation

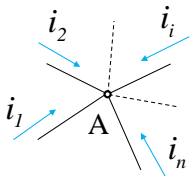
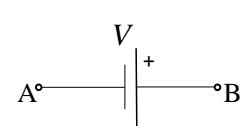
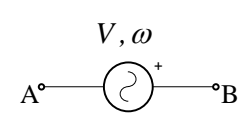
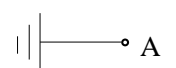
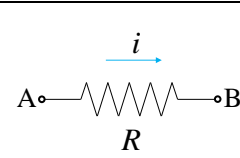
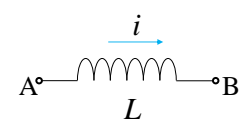
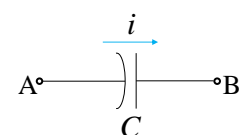
Next, we look at electrical actuation systems. Physically, dynamic simulation of an electrical actuation system is mainly characterized by:

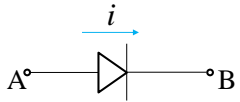
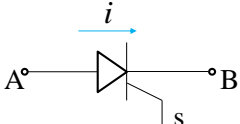
- acceleration of the mechanical system
- dynamics of electrical components

Often, the internal dynamics of electrical machines has much higher eigenfrequencies than that of the mechanical system. Because of this, it is typically enough to handle the steady state characteristics of the electrical drives. In this section we will investigate the dynamics of some basic components and set up the governing equations for a DC (direct current) motor and an AC (alternating current) induction motor.

In Tab. 5.2 the governing equations are listed for most typical electrical components.

Table 5.2 Governing equations for basic electrical components

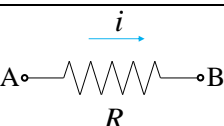
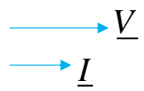
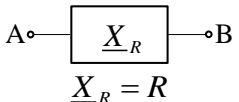
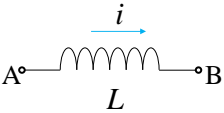
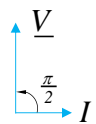
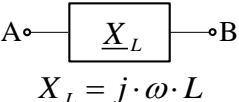
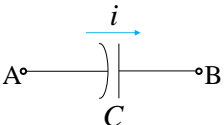
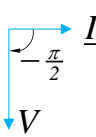
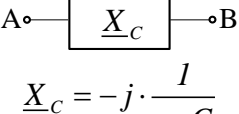
Description	Symbol	Equations
Voltage node		$\sum_{i=1}^n i_i = 0$
DC voltage source		$v_B - v_A = V$
AC voltage source		$v_B - v_A = \sqrt{2} \cdot V \cdot \sin(\omega \cdot t)$
Ground		$v_A = 0$
Resistance		$v_A - v_B = i \cdot R$
Inductance		$\frac{di}{dt} = \frac{1}{L} \cdot (v_A - v_B)$
Capacitance		$\frac{d(v_A - v_B)}{dt} = \frac{1}{C} \cdot i$

Diode		$i = 0 \quad v_A \leq v_B$ $v_A = v_B \quad i > 0$
Transistor		$i = 0 \quad v_A \leq v_B \quad \text{OR} \quad s = \text{OFF}$ $v_A = v_B \quad i > 0 \quad \text{AND} \quad s = \text{ON}$

In Table 5.2 the diode and transistor models are ideal model which means that they have two operational modes, either closed or open. The diode simply allows for free current passage from A to B and blocks from B to A. The transistor works similarly but requires an external signal to be ON in order to allow the free current passage.

The differential equations are introduced via the inductance and capacitance equations. For an inductance the state variable is the inductance current, i.e., the current in an inductance must be given an initial value in time integration. Similarly, the voltage across a capacitor is a state variable and must be provided an initial value in simulations. As mentioned earlier, it is quite often the case that the dynamics of the electrical system can be ignored. In that case the equations related to the electronics reduce to algebraic equations that describe the steady state relations. A special branch of steady state computations on electronic systems is that of systems subjected to AC current or voltage sources. For such systems all currents and voltages will, after the transients has died out, oscillate with the source frequency. The steady state behavior of individual voltages and currents is not needed as time series but only as an amplitude and a relative phase for each parameter. Steady state analysis of AC-circuits can be reduced to purely algebraic analysis where any resistance, impedance and conductance can be handled as imaginary frequency dependent resistances normally referred to as impedances, see Table 5.3.

Table 5.3 AC steady state equations for basic electrical components.

Base component current source: $i = \sqrt{2} \cdot I \cdot \sin(\omega \cdot t)$	Voltage amplitude and phase: $v_A - v_B = \sqrt{2} \cdot V \cdot \sin(\omega \cdot t + \varphi)$	Imaginary formulation: $\underline{V} = V \cdot \cos \varphi + j \cdot V \cdot \sin \varphi$ $\underline{I} = I$	Impedance (resistance): $\underline{V} = \underline{X} \cdot \underline{I}$
	$V = R \cdot I$ $\varphi = 0$		 $\underline{X}_R = R$
	$V = \omega \cdot L \cdot I$ $\varphi = \frac{\pi}{2}$		 $\underline{X}_L = j \cdot \omega \cdot L$
	$V = \frac{I}{\omega \cdot C}$ $\varphi = -\frac{\pi}{2}$		 $\underline{X}_C = -j \cdot \frac{I}{\omega \cdot C}$

Notice the use of the so-called RMS-values, $I_{RMS} = I$ and $V_{RMS} = V$, rather than the actual amplitude of the voltages and currents (which are $\sqrt{2}$ times larger). This is traditional within analysis of

electrical systems and simply relates to the computation of the average power dissipated by a resistance across a cycle:

$$P = \frac{1}{T} \cdot \int_0^T i \cdot v \cdot dt = \frac{\omega}{2 \cdot \pi} \cdot \int_0^{\frac{2\pi}{\omega}} i^2 \cdot R \cdot dt = \frac{\omega \cdot R}{2 \cdot \pi} \cdot \int_0^{\frac{2\pi}{\omega}} \left[\sqrt{2} \cdot I \cdot \sin(\omega \cdot t) \right]^2 \cdot dt =$$

$$\frac{\omega \cdot R \cdot I^2}{\pi} \cdot \int_0^{\frac{2\pi}{\omega}} \sin^2(\omega \cdot t) \cdot dt = \frac{\omega \cdot R \cdot I^2}{\pi} \cdot \left[\frac{t}{2} - \frac{\sin(2 \cdot \omega \cdot t)}{4 \cdot \omega} \right]_0^{\frac{2\pi}{\omega}} = R \cdot I^2 \quad (5.11)$$

From (5.11) it is clear that the RMS-value of the current can be used directly to compute the power dissipation. In similar fashion, it can be shown that:

$$P = \frac{V^2}{R} \quad (5.12)$$

It is customary to do the steady state AC-circuit analysis on RMS-values rather than total values. As an example, a typical household voltage refers to the RMS-value of 220V but this actually corresponds to an amplitude of $\approx 311V$ or, indeed, a peak-to-peak voltage of $\approx 622V$.

The main modeling advantage of the impedances is that any circuit can be assembled to an effective impedance in the same way that a network of resistors can be assembled to an effective resistor. The two basic rules used in such operations refer to resistances/impedances in either series or parallel connection, see Fig. 5.4 and Fig. 5.5.

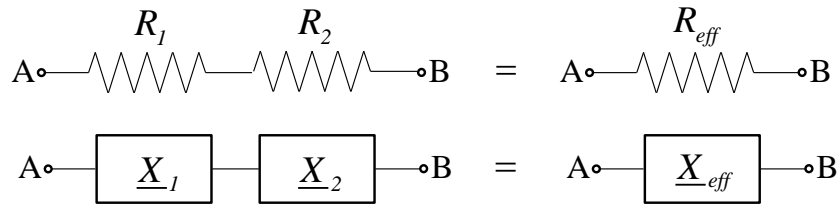


Figure 5.4 Two resistances and impedances in series reduced to an effective resistance and impedance, respectively.

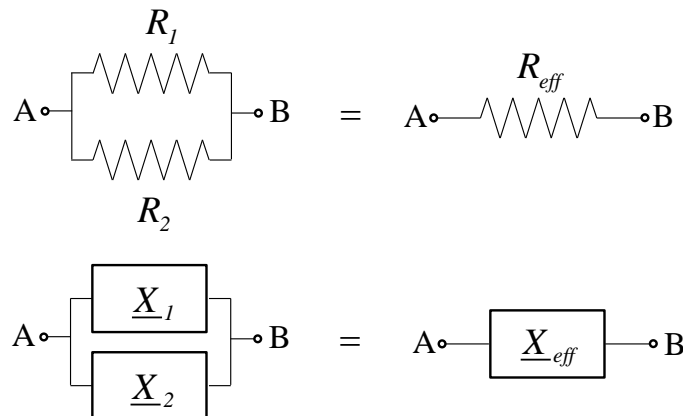


Figure 5.5 Two resistances and impedances in parallel reduced to an effective resistance and impedance, respectively.

The equations for the effective resistance/impedance series connections are:

$$R_{eff} = R_1 + R_2 \quad \underline{X}_{eff} = \underline{X}_1 + \underline{X}_2 \quad (5.13)$$

The equations for the effective resistance/impedance parallel connections are:

$$R_{eff} = \frac{I}{\frac{I}{R_1} + \frac{I}{R_2}} \quad \underline{X}_{eff} = \frac{I}{\frac{I}{\underline{X}_1} + \frac{I}{\underline{X}_2}} \quad (5.14)$$

Two of the most popular motors in engineering are the permanent magnet DC motor and the three-phase AC induction motor. In the following the governing equations for the torque produced by these motors as a function of angular velocity of the rotor is derived from equivalent circuits.

The permanent magnet DC motor is composed of a permanent magnet embedded in the stator where it generates an approximately constant magnetic field across the rotor-stator air gap of the motor. The power supply is a DC voltage that is commuted to the rotor windings by means of brush rings. In Fig. 5.6 the equivalent circuit for a permanent magnet DC motor is shown.

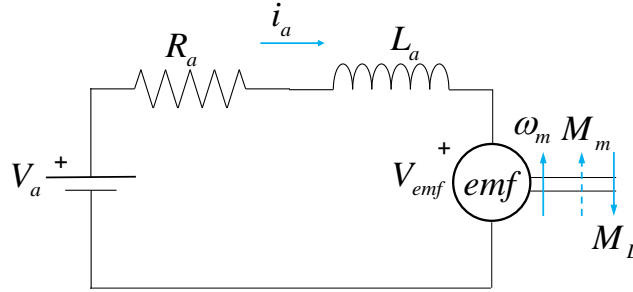


Figure 5.6 Equivalent circuit for a permanent magnet DC motor.

The indices a refer to the armature (rotor). The supply voltage, V_a , can be either constant or continuously adjusted with a view to control the speed of the motor. The rotor resistance, R_a , represents the ohmic resistance of the windings and current induced voltage losses in the brushes. The rotor inductance, L_a , reflects the self-inductance of the rotor winding. An electro motoric force, V_{emf} , that is proportional to the rotor speed is introduced in the rotor windings. The angular velocity of the motor is referred to as ω_m and the torque generated by the motor on the rotor is M_m . The external load, M_L , is considered positive when acting in the opposite direction of the rotor motion and negative when acting in the same direction. In steady state, we have $M_m = M_L$, however, in general the differential equation for the motor is:

$$J_{eff} \cdot \dot{\omega}_m = M_m - M_L \quad (5.15)$$

In (5.15) we use the effective mass moment of inertia of the drive train rigidly connected to the rotor. The DC motor is characterized by a motor constant, K_m , that describes the correlation between armature current and motor torque as well as the induced electro motoric force and the rotor speed:

$$M_m = K_m \cdot i_a \quad (5.16)$$

$$V_{emf} = K_m \cdot \omega_m \quad (5.17)$$

The armature current is computed in two different ways depending on the modeling of the DC motor. If the electrical circuit is modeled dynamically the rotor inductance is taken into account:

$$V_a = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt} + K_m \cdot \omega_m \Rightarrow$$

$$\frac{di_a}{dt} = \frac{1}{L_a} \cdot (V_a - R_a \cdot i_a - K_m \cdot \omega_m) \quad (5.18)$$

Eqs. (5.15) and (5.18) are the differential equations used to compute \dot{i}_a and $\dot{\omega}_m$. This means that both armature current and motor speed are state variables that must be initialized.

If the dynamics of the armature is neglected, there is only one differential equation: (5.15). The circuit equations reduces to:

$$V_a = R_a \cdot i_a + K_m \cdot \omega_m \quad (5.19)$$

Combining (5.16) and (5.19) yields the following steady state relationship between motor torque and motor speed, see also Fig. 5.7:

$$M_m = M_{ST} \cdot \left(1 - \frac{\omega_m}{\omega_{NL}} \right) \quad (5.20)$$

In (5.20) M_{ST} is the stall torque which corresponds to the torque that will stall the motor, i.e., $\omega_m = 0$. The no-load speed, ω_{NL} , is the angular velocity of the rotor without any external load, $M_m = M_L = 0$. The expression for the two parameters are:

$$M_{ST} = \frac{V_a \cdot K_m}{R_a} \quad (5.21)$$

$$\omega_{NL} = \frac{V_a}{K_m} \quad (5.22)$$

The rated power, P_r , for a DC motor is given as the product of the rated speed, ω_r , and rated torque, M_r , and it represents a point of operation where the motor can work continuously without overheating.

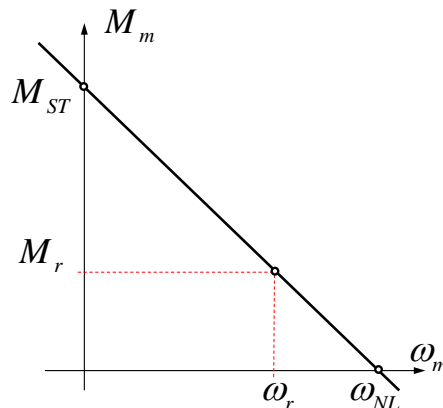


Figure 5.7 Steady state characteristics of a permanent magnet DC motor.

The maximum power that the motor can deliver is at $\omega_m = 0.5 \cdot \omega_{NL}$ but only for a limited time interval.

$$P_r = M_r \cdot \omega_r \quad (5.23)$$

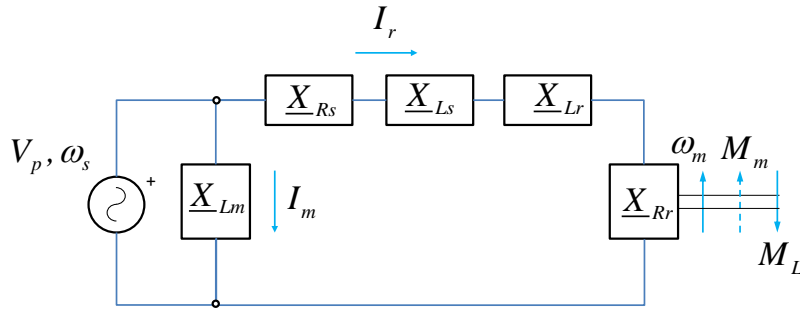
$$P_{max} = 0.25 \cdot M_{ST} \cdot \omega_{NL} \quad (5.24)$$

The easily most widespread motor in the world is the three-phase short circuited induction motor. This type of motor relies on the availability of three AC voltage sources (lines) each with an RMS value of V_L . The rotor is short circuited, i.e., there is no physical connection between the stator and rotor. The stator windings are divided into three phases that receive their supply voltage from the AC source lines. Depending on the way the AC source lines are connected (either in star or delta) each phase of the motor will be subjected to an AC voltage source with an RMS value, V_p , of either: $V_p = \sqrt{3} \cdot V_L$ (star) or $V_p = V_L$ (delta). The three phases lie with a phase offset of 120° and this facilitates that the stator windings can generate a magnetic field of constant strength that rotates with an angular velocity of:

$$\omega_s = \frac{2}{p} \cdot 2 \cdot \pi \cdot f_s \quad (5.25)$$

$$n_s = \frac{2}{p} \cdot f_s \quad (5.26)$$

In (5.25) ω_s is the synchronous angular velocity, n_s is the synchronous speed, p is the number of magnetic poles (twice the number of coils made of each phase) and f_s is the frequency of the voltage source. In Europe, the commercially available AC net has a frequency of $f_s = 50 \text{ Hz}$. The most popular types of AC motors have either two poles, $p = 2$, or four poles, $p = 4$, which gives a synchronous speed of $n_s = 3000 \frac{\text{rev}}{\text{min}}$ and $n_s = 1500 \frac{\text{rev}}{\text{min}}$, respectively. Traditionally, an induction motor is analyzed by means of a steady state AC-circuit representing a single phase, see Fig. 5.8.

**Figure 5.8** Equivalent circuit for a single steady state phase of an AC induction motor.

The circuit in Fig. 5.8 introduces a large number of simplifications that, in general, are valid for most commercially available AC induction motors. Basically, the circuit models the current that is used to magnetize the air gap between rotor and stator, I_m , and the current that is induced in the rotor is represented by I_r . The air gap inductance is represented by $\underline{X}_{Lm} = j \cdot \omega_s \cdot L_m$, the resistance of the stator winding is represented by $\underline{X}_{Rs} = R_s$, and the leakage inductance in the stator and rotor windings

are represented by $\underline{X}_{Ls} = j \cdot \omega_s \cdot L_s$ and $\underline{X}_{Lr} = j \cdot \omega_s \cdot L_r$, respectively. Finally, the impedance \underline{X}_{Rr} needs extra discussion since it represents two power outlets. Firstly, the total power that crosses the air gap between the stator and the rotor, $P_{S \rightarrow R}$, can be written as:

$$P_{S \rightarrow R} = 3 \cdot \frac{R_r}{s} \cdot I_r^2 \quad (5.27)$$

The derivation of (5.27) lies outside the scope of these notes, however, the factor of three simply reflects that there are three phases. In (5.27) the resistance of the short-circuited rotor windings, R_r , is introduced together with the dimensionless slip parameter, s , that is defined in the following way:

$$s = \frac{\omega_s - \omega_m}{\omega_s} \quad (5.28)$$

Hence, the slip is, $s = 0$, when the rotor is rotating with the same speed as the magnetic field generated by the stator windings. When the rotor is stalled the slip is $s = 1$, and if the motor is pulled into generating mode by a negative load, $\omega_m > \omega_s$, the slip becomes negative, $s < 0$. The power that is divided between dissipation in the rotor resistance, P_{Rr} , and mechanical power on the rotor, P_m . They can be computed as:

$$P_{Rr} = 3 \cdot R_r \cdot I_r^2 \quad (5.29)$$

$$P_m = P_{S \rightarrow R} - P_{Rr} = 3 \cdot \frac{(1-s)}{s} \cdot R_r \cdot I_r^2 \quad (5.30)$$

Electrically, this corresponds to two resistances in series with the values R_r and $\frac{(1-s)}{s} \cdot R_r$, respectively. When added together according to (5.13) we obtain an effective resistance/impedance that represents power dissipation in the resistance of the rotor and power transmission to the mechanical load:

$$\underline{X}_{Rr} = \frac{R_r}{s} \quad (5.31)$$

An expression for the steady state value of the rotor current can be developed using (5.13) to combine all the impedances in the stator-rotor line:

$$\underline{I}_r = \frac{V_p}{R_s + \frac{R_r}{s} + j \cdot \omega_s (L_s + L_r)} \quad (5.32)$$

The torque produced by the motor can be found from basic mechanics combined with (5.28) and (5.30) as:

$$M_m = \frac{P_m}{\omega_m} = \frac{3 \cdot \frac{(1-s)}{s} \cdot R_r \cdot I_r^2}{(1-s) \cdot \omega_s} = \frac{3 \cdot R_r \cdot I_r^2}{s \cdot \omega_s} \quad (5.33)$$

Combining (5.32) and (5.33) yields the following expression for the motor torque:

$$M_m = \frac{3 \cdot R_r \cdot V_p^2}{s \cdot \omega_s \cdot \left[\left(R_s + \frac{R_r}{s} \right)^2 + \omega_s^2 \cdot (L_s + L_r)^2 \right]} \quad (5.34)$$

The differential equation for the AC induction motor is the same as that for the DC permanent magnet motor (or indeed any motor producing a torque M_m) simply (5.15) with the motor torque computed from (5.34). As long as the phase voltage and line frequency are not changed the only parameter in (5.15) that varies during motor operation is the slip. It is, however, easily computed from the state variable ω_m . The no-load speed of the motor is $\omega_{NL} = \omega_s$. Whenever the motor reaches that the no-load speed the slip becomes zero, $s = 0$, which will cause numerical problems in (5.34) unless some if-then-else logic is added. As an example (5.28) could be augmented to:

$$s = \begin{cases} s^* & |s^*| \geq s_\varepsilon \\ s_\varepsilon & s_\varepsilon > s^* \geq 0 \\ -s_\varepsilon & 0 > s_\varepsilon > -s^* \end{cases} \quad (5.35)$$

$$s^* = \frac{\omega_s - \omega_m}{\omega_s} \quad (5.36)$$

In (5.35) $s_\varepsilon \approx 10^{-6} \dots 10^{-8}$ is simply a numerically very small slip value that is introduced to avoid division by zero. Eq. (5.34) can be used to plot the steady state torque-speed characteristics of the AC induction motor. For typical values of the different parameters we get a curve as shown in Fig. 5.9.

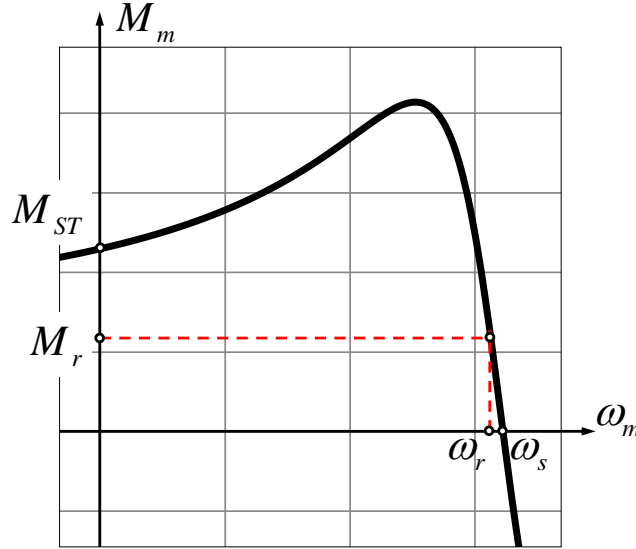


Figure 5.9 Steady state characteristics of an induction AC motor.

In Fig. 5.9 we find the rated torque and speed that yields the rated power of the motor according to (5.23). As in the case of the DC motor, this is not the maximum power that the AC motor can deliver, however, it is the power that it can deliver continuously without overheating.

CHAPTER 6: Numerical Methods

Time domain simulation is, basically, the solution of differential equations by means of a numerical method called time integration. There are a number of other numerical methods which all are characterized by the use of numbers and, normally, many algebraic operations on said numbers in order to solve a set of equations. Because of the many algebraic operations, numerical methods are normally used together with computers in order to get the job done within reasonable time and effort. In this chapter we will briefly review the concept of matrices and algebraic vectors and then look at numerical methods for solving a set of linear equations and a set of non-linear equations, respectively. Also, we will look at numerical techniques for parameter (or system) identification and minimization (or optimization).

6.1 Matrices

A matrix is a two-dimensional array of numbers consisting of n rows and m columns:

$$\underline{\underline{A}} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \quad (6.1)$$

Hence, the matrix has a total number of $n \cdot m$ components (or entries) and the component of the i 'th row and j 'th column is referred to as:

$$\underline{\underline{A}}_{[i,j]} = a_{i,j} \quad (6.2)$$

The transposed of $\underline{\underline{A}}$ will have m rows and n columns. It is defined as

$$b_{i,j} = a_{j,i} \quad (6.3)$$

$$\underline{\underline{B}} = \underline{\underline{A}}^T = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{n,1} \\ \vdots & & \vdots \\ a_{1,m} & \cdots & a_{n,m} \end{bmatrix} \quad (6.4)$$

A square matrix, $n = m$, can be inverted if it is not singular which corresponds to having the determinant equal to zero. Normally, inverting a $m = 1$ square matrix with $n \geq 3$ is a tedious exercise and should preferably be done numerically by using existing routines or procedures of some existing software.

An algebraic vector is a special version of a matrix with either $m = 1$ (a column vector) or $n = 1$ (a row vector).

$$\underline{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \quad \underline{a} = [a_1 \quad \cdots \quad a_m] \quad (6.5)$$

Finally, a scalar may be thought of as a square matrix with $n = m = 1$.

Two matrices can be added or subtracted if they have the same number of rows and columns:

$$\underline{\underline{C}}^{(n \times m)} = \underline{\underline{A}}^{(n \times m)} \pm \underline{\underline{B}}^{(n \times m)} \quad c_{i,j} = a_{i,j} \pm b_{i,j} \quad (6.6)$$

Two matrices can be multiplied, however, the order of the factors is not insignificant and the number of columns in the first matrix of the product must correspond to the number of rows in the second matrix:

$$\underline{\underline{C}}^{(n \times p)} = \underline{\underline{A}}^{(n \times m)} \cdot \underline{\underline{B}}^{(m \times p)} \quad c_{i,j} = \sum_{k=1}^m a_{i,k} \cdot b_{k,j} \quad (6.7)$$

In Tab. 6.1 a number of examples of matrix manipulation in Matlab is given.

Table 6.1 Matrix manipulations in Matlab.

<p>Create a matrix and then access element in 3rd row and 2nd column:</p> <pre>>> A=[2 -3 1 ; 5 2 0 ; -4 6 5]</pre> <p>A =</p> <pre> 2 -3 1 5 2 0 -4 6 5 >> A(3,2)</pre> <p>ans =</p> <pre> 6</pre>	<p>Multiply two matrices $\underline{\underline{C}} = \underline{\underline{A}} \cdot \underline{\underline{B}}$</p> <pre>>> A=[2 4 4 ; 5 6 9]</pre> <p>A =</p> <pre> 2 4 4 5 6 9 >> B=[2 3; 6 6; 0 9]</pre> <p>B =</p> <pre> 2 3 6 6 0 9 >> C=A*B</pre> <p>C =</p> <pre> 28 66 46 132</pre>
<p>Multiply two vectors as matrices; (dot product) $c = \underline{a} \cdot \underline{b}$</p> <pre>>> a=[1 4 7]</pre> <p>a =</p> <pre> 1 4 7 >> b=[0 ; -3 ; 2]</pre> <p>b =</p> <pre> 0 -3 2 >> c=a*b</pre> <p>c =</p> <pre> 2</pre>	<p>Multiply two vectors element wise $\underline{c} = \underline{a} \cdot \underline{b}$</p> <pre>>> a=[3 ; 5 ; 2]</pre> <p>a =</p> <pre> 3 5 2 >> b=[1 ; 0 ; -4]</pre> <p>b =</p> <pre> 1 0 -4 >> c=a.*b</pre> <p>c =</p> <pre> 3 0 -8</pre>
<p>Transpose a matrix $\underline{\underline{B}} = \underline{\underline{A}}^T$</p>	<p>Invert a matrix $\underline{\underline{B}} = \underline{\underline{A}}^{-1}$</p>

<pre>>> A=[2 4 4 ; 5 6 9]</pre> <p>A =</p> <pre> 2 4 4 5 6 9 </pre> <pre>>> B=A'</pre> <p>B =</p> <pre> 2 5 4 6 4 9 </pre>	<pre>>> A=[1 2 3 ; -1 4 4 ; 0 3 -8]</pre> <p>A =</p> <pre> 1 2 3 -1 4 4 0 3 -8 </pre> <pre>>> B=inv(A)</pre> <p>B =</p> <pre> 0.6377 -0.3623 0.0580 0.1159 0.1159 0.1014 0.0435 0.0435 -0.0870 </pre>
---	--

6.2 Linear Equations

A set of n linear equations can be written as:

$$\underline{\underline{A}} \cdot \underline{x} = \underline{y} \quad (6.8)$$

where $\underline{\underline{A}}^{(n \times n)}$ is the coefficient matrix, $\underline{x}^{(n \times 1)}$ is the column vector of unknowns and $\underline{y}^{(n \times 1)}$ is the column vector of the right-hand side. The solution to (6.8) is simply:

$$\underline{x} = \underline{\underline{A}}^{-1} \cdot \underline{y} \quad (6.9)$$

Example 2

Let us consider a simple ideal mechanical pendulum from example 1.

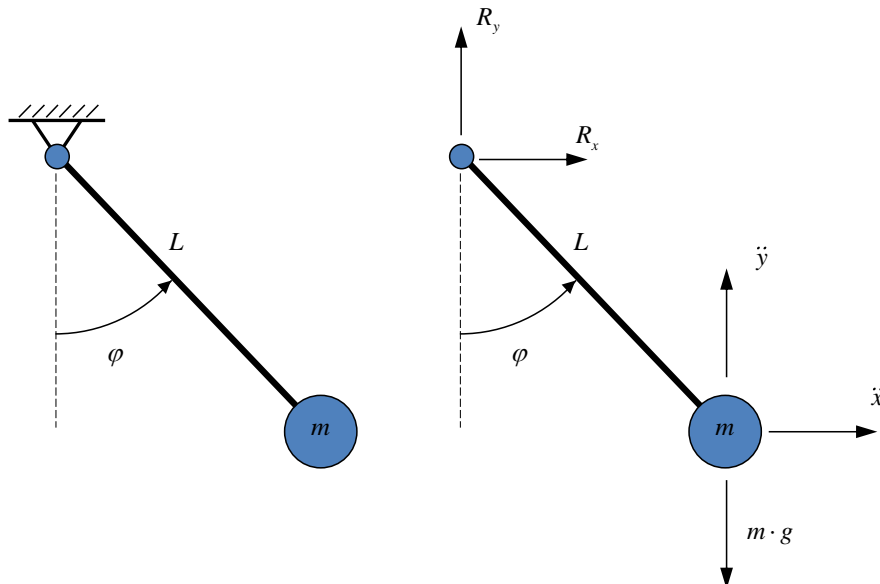


Figure 6.1 Simple pendulum with concentrated mass.

The governing differential and algebraic equations for this pendulum is:

$$m \cdot \ddot{x} = R_x \quad (6.10)$$

$$m \cdot \ddot{y} = R_y - m \cdot g \quad (6.11)$$

$$\ddot{\varphi} = -\frac{g}{L} \cdot \sin \varphi \quad (6.12)$$

$$x = L \cdot \sin \varphi \Rightarrow \ddot{x} = L \cdot \cos \varphi \cdot \ddot{\varphi} - L \cdot \sin \varphi \cdot \dot{\varphi}^2 \quad (6.13)$$

$$y = -L \cdot \cos \varphi \Rightarrow \ddot{y} = L \cdot \sin \varphi \cdot \ddot{\varphi} + L \cdot \cos \varphi \cdot \dot{\varphi}^2 \quad (6.14)$$

Knowing the state variables φ and $\dot{\varphi}$ we have five unknowns: $\underline{x} = [\ddot{x} \quad \ddot{y} \quad \ddot{\varphi} \quad R_x \quad R_y]$ and it is possible to set up the equations according to (6.8):

$$\underbrace{\begin{bmatrix} m & 0 & 0 & -1 & 0 \\ 0 & m & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -L \cdot \cos \varphi & 0 & 0 \\ 0 & 1 & -L \cdot \sin \varphi & 0 & 0 \end{bmatrix}}_{\underline{A}} \cdot \underbrace{\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\varphi} \\ R_x \\ R_y \end{bmatrix}}_{\underline{x}} = \underbrace{\begin{bmatrix} 0 \\ -m \cdot g \\ -\frac{g}{L} \cdot \sin \varphi \\ -L \cdot \sin \varphi \cdot \dot{\varphi}^2 \\ L \cdot \cos \varphi \cdot \dot{\varphi}^2 \end{bmatrix}}_{\underline{y}} \quad (6.15)$$

Setting the initial rotation to 45° and starting from rest we can solve for the five unknowns in Matlab using the following script:

```
close all;
clear;
%Basic data
m=2.0;
g=9.81;
L=1.0;
phi=pi/4;
phiDot=0;
%Set up coefficient matrix
M=[m 0 0 -1 0 ; ...
    0 m 0 0 -1; ...
    0 0 1 0 0; ...
    1 0 -L*cos(phi) 0 0; ...
    0 1 -L*sin(phi) 0 0];
%Set up right hand side as a column vector
y=[0 -m*g -g/L*sin(phi) -L*sin(phi)*phiDot^2 -L*cos(phi)*phiDot^2]';
%Solve for x
x=inv(M)*y
```

yielding the following solution:

```
x =
-4.9050
-4.9050
-6.9367
-9.8100
 9.8100
```

Setting up the coefficient matrix, the right hand side and solving for the unknowns could easily be carried out inside a time loop of a time domain simulation program. In that case, not only the position, velocity and acceleration of the system, but also the reactive forces would be computed for each time step.

6.3 Nonlinear equations

With nonlinear equations we cannot write the equation system in the nice way of (6.8) but has to use a more general formulation:

$$\underline{\Psi}(\underline{x}) = \underline{0}$$

$$\begin{bmatrix} \Psi_1(\underline{x}) \\ \vdots \\ \Psi_n(\underline{x}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad (6.16)$$

In general, nonlinear equations must be solved iteratively. Initially, let us examine a single nonlinear function with a single unknown:

$$\Psi(x) = 0 \quad (6.17)$$

To solve it we use a Taylor expansion around some initial value for x that we assume lie close to the solution. Denoting the initial value (or initial guess) x_0 we get for a first order expansion:

$$\Psi^*(x) = \Psi(x_0) + \Psi'(x_0) \cdot (x - x_0) \quad (6.18)$$

We solve (6.17) and get an expression for x :

$$\Psi^*(x) = \Psi(x_0) + \Psi'(x_0) \cdot (x - x_0) = 0 \Rightarrow x = x_0 - \frac{\Psi(x_0)}{\Psi'(x_0)} \quad (6.19)$$

Due to the linearization, the new value of x is not a final solution to (6.17) but it is probably a better estimate than x_0 , and the procedure outlined in (6.19) is simply repeated until an acceptable accuracy is obtained. This iterative procedure is called Newton-Raphson iteration, and it can be written in the following way:

$$x^{(i+1)} = x^{(i)} - \frac{\Psi(x^{(i)})}{\Psi'(x^{(i)})} \quad (6.20)$$

In (6.20) the superscripts indicate the i 'th and $i+1$ 'th estimate respectively.

Example 3

As an example, let us find a solution to the non-linear equation:

$$\Psi(x) = 2 \cdot x^2 + 7 \cdot x + 3 = 0 \quad (6.21)$$

Eq. (6.21) belongs to a family of non-linear equations referred to as second order equations. These non-linear equations are so common and simple that analytical solutions are easily obtained as $x = -0.5$ or $x = -3$. Using Newton-Raphson we must come up with an initial guess and then follow (6.20). In Table 6.2 the progress of the Newton-Raphson can be followed:

Table 6.2 Newton-Raphson iteration history. Error tolerance is $|\Psi(x)| < 10^{-5}$.

Iteration count, i	x	$\Psi(x) = 2 \cdot x^2 + 7 \cdot x + 3$	$\Psi'(x) = 4 \cdot x + 7$
1	0	3	7
2	-0.4286	0.3673	5.2857
3	-0.4981	0.0097	5.0077
4	-0.5000	0.0000	5.0000

Using a $x=0$ for an initial guess yielded one of the two possible solutions. In Table 6.3 we can follow the history for an initial guess of $x=-10$ leading to the other possible solution.

Table 6.3 Newton-Raphson iteration history. Error tolerance is $|\Psi(x)| < 10^{-5}$.

Iteration count, i	x	$\Psi(x) = 2 \cdot x^2 + 7 \cdot x + 3$	$\Psi'(x) = 4 \cdot x + 7$
1	-10	133	-33
2	-5.9697	32.4867	-16.8788
3	-4.0450	7.4090	-9.1800
4	-3.2379	1.3028	-5.9516
5	-3.0190	0.0958	-5.0761
6	-3.0001	0.0000	-5.0006

The Newton-Raphson is not fool proof. If we start from $x = -\frac{7}{4}$ the procedure crashes because of division by zero. If we start too far away from a solution, the procedure may diverge rather than converge. Finally, the procedure may converge, but to a different solution than the one we are looking for.

The Newton-Raphson iteration is already embedded in Matlab as a standard procedure referred to as: `fsolve`. This procedure needs a script that computes the value of the residuals $\underline{\Psi}$ as a function of the unknowns \underline{x} and also an initial guess for the unknowns \underline{x}_0 .

The following Matlab code does the same as explained in Table 6.2:

```
close all;
clear;
%Initial guess
x0=0;
%Solve
x=fsolve('Residual_Example3_MAS407',x0)
```

returns the following result:

```
Equation solved.
fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.
<stopping criteria details>
```

```
x =

-0.5000
```

The equation to be solved is written in the Matlab script: *Residual_Example3_MAS407.m* and it simply looks like this:

```
function f = Residual_Example3_MAS407(x)
f=2*x^2+7*x+3;
```

For a set for non-linear equations the Newton-Raphson procedure of (6.20) becomes:

$$\underline{x}^{(i+1)} = \underline{x}^{(i)} - \underline{\Psi}_x^{-1}(\underline{x}^{(i)}) \cdot \underline{\Psi}(\underline{x}^{(i)}) \quad (6.22)$$

where the Jacobian matrix is given as:

$$\underline{\Psi}_x = \frac{\partial \underline{\Psi}}{\partial \underline{x}} \quad \underline{\Psi}_{x[i,j]} = \frac{\partial \Psi_i}{\partial x_j} \quad (6.23)$$

Example 4

If we want to solve for the intersection point between two circles, see Fig. 6.2, then we have two equations and two unknowns.

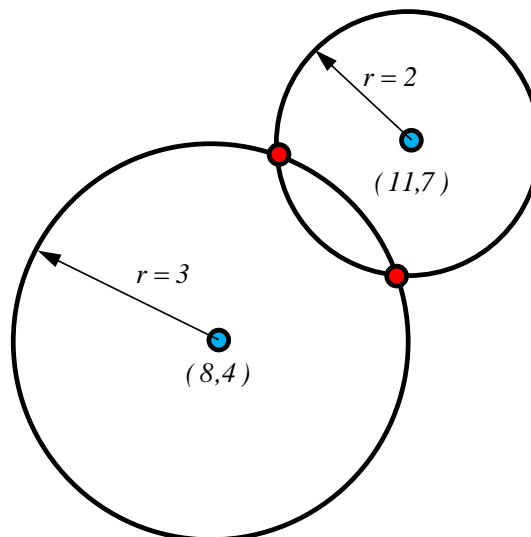


Figure 6.2 Two circles intersecting each other.

In Matlab we solve this in the following way:

```
close all;
clear;
%Initial guess
x0=[0 0]';
%Solve
x=fsolve('Residual_Example4_MAS407',x0)
```

returns the following result:

```
Equation solved.
fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.
<stopping criteria details>
```

```
x =
    9.0076
    6.8257
```

The equation to be solved is written in the Matlab script: *Residual_Example3_MAS407.m* and it simply looks like this:

```
function f = Residual_Example4_MAS407(x)
f(1)=(x(1)-8)^2+(x(2)-4)^2-3^2;
f(2)=(x(1)-11)^2+(x(2)-7)^2-2^2;
```

A different initial guess, for example

```
close all;
clear;
%Initial guess
x0=[10 0]';
%Solve
x=fsolve('Residual_Example4_MAS407',x0)
```

yields the other solution:

```
x =
   10.8257
    5.0076
```

6.4 System Identification

In many situations measurements on a given system are available. At the same time it is desirable to set up a model of the system that is in accordance with the measurements with a view to conduct further experiments using the model rather than the physical system. In order to obtain a model that is in accordance with the measurements it is necessary to do some identification of the parameters of the system. In the following we will assume that we can set up the following linear equations for our unknown parameters:

$$\underline{\tilde{\Phi}} \cdot \underline{p} = \underline{\tilde{Y}} \quad (6.24)$$

In (6.24) \underline{p} are the unknown parameters that we want to identify. The right-hand side is $\underline{\tilde{Y}}$ and is made up of known and measured values. The coefficient matrix $\underline{\tilde{\Phi}}$ also comes from measured values. Normally, $\underline{\tilde{Y}}$ and $\underline{\tilde{\Phi}}$ will contain a high number of rows corresponding to each sampling of measured data. The number of columns in $\underline{\tilde{\Phi}}$ corresponds to the number of unknown parameters, i.e., the number of rows in \underline{p} . Therefore, $\underline{\tilde{\Phi}}$ is not a square matrix and it cannot be inverted to yield the parameters. In short, we have more equations than unknowns. What we do, is simply to multiply both side of (6.24) with the transposed of $\underline{\tilde{\Phi}}$, thereby creating a square coefficient matrix on \underline{p} that can be inverted:

$$\underline{\tilde{\Phi}} \cdot \underline{p} = \underline{\tilde{Y}} \Rightarrow \underline{\tilde{\Phi}}^T \cdot \underline{\tilde{\Phi}} \cdot \underline{p} = \underline{\tilde{\Phi}}^T \cdot \underline{\tilde{Y}} \Rightarrow \underline{p} = \left(\underline{\tilde{\Phi}}^T \cdot \underline{\tilde{\Phi}} \right)^{-1} \cdot \underline{\tilde{\Phi}}^T \cdot \underline{\tilde{Y}} \quad (6.25)$$

As long as the problem is linear, i.e., can be formulated as (6.24) we can use (6.25) to determine the unknown parameters \underline{p} , from the measured data $\underline{\tilde{Y}}$ and $\underline{\tilde{\Phi}}$.

Example 5

As an example, assume that we have measurements of the position, velocity and acceleration of the system shown in Fig. 6.3.

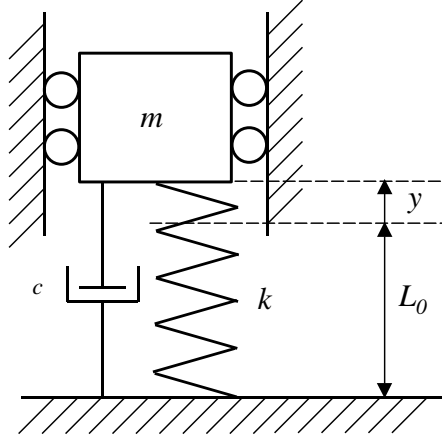


Figure 6.3 Mass-spring-damper system.

In that case we have three arrays of measurements: $\underline{\tilde{y}}$, $\underline{\dot{\tilde{y}}}$ and $\underline{\ddot{\tilde{y}}}$, and we know the mass, m , but not the damper, b , and the spring, k . In that case we can establish (6.24) as follows:

$$\begin{bmatrix} -\underline{\tilde{y}} & -\underline{\dot{\tilde{y}}}] \cdot \begin{bmatrix} k \\ b \end{bmatrix} = m \cdot \underline{\ddot{\tilde{y}}} \quad (6.26)$$

And the spring and damper constants can be identified from:

$$\begin{bmatrix} k \\ b \end{bmatrix} = \left(\begin{bmatrix} -\underline{\tilde{y}} & -\underline{\dot{\tilde{y}}}]^T \cdot \begin{bmatrix} -\underline{\tilde{y}} & -\underline{\dot{\tilde{y}}}] \right)^{-1} \cdot \begin{bmatrix} -\underline{\tilde{y}} & -\underline{\dot{\tilde{y}}}]^T \cdot m \cdot \underline{\ddot{\tilde{y}}} \quad (6.27)$$

Using the matrix manipulation routines of Matlab this is easily done in one or two script lines.

6.5 Minimization

During model based design in practical design work it is customary to work with several criteria. These criteria may be divided into subjects for minimization, equality constraints and inequality constraints. As an example, the three different types of criteria may be recognized in this idealized design specification of a robotic arm:

- the price should be minimized
- the motors must be electrical and equipped with a gearbox
- the manipulator must have a total of four degrees-of-freedom
- the manipulator must have a reach of a specified radius

- the weight must be smaller than a specified maximum mass
- the accuracy of the tool point must be smaller than a specified tolerance

These type of problems are very suitable for a numerical approach and there exist a wide variety of algorithms for the systematic manipulation of a set of design variables with a view to minimize some objective while meeting both the inequality and the equality constraints.

Mathematically, this can be formulated in a standardized way as:

$$\begin{aligned} &\text{minimize} && O(\underline{y}) \\ &\text{subject to} && \underline{g}(\underline{y}) \leq \underline{0} \\ &&& \underline{h}(\underline{y}) = \underline{0} \end{aligned} \quad (6.28)$$

The main target of the minimization is find to the set of design variables, $\underline{y} = [y_1 \cdots y_m]$ that yields the minimum value of the objective function, O . In most practical problems, there will be a number of inequality constraints on the design variables, $\underline{g} = [g_1(\underline{y}) \cdots g_n(\underline{y})]$, and (although less common) a number of equality constraints, $\underline{h} = [h_1(\underline{y}) \cdots h_p(\underline{y})]$. The design variables span an m-dimensional space, and the part of this design space that fulfills all constraints is called the feasible design space. Any valid solution to (6.28) must be found within the feasible design space.

Example 6

Find the minimum value of the function $O = x \cdot y$, if $\underline{y} = [x \ y]$ must lie within the feasible design space shown in Fig. 6.4:

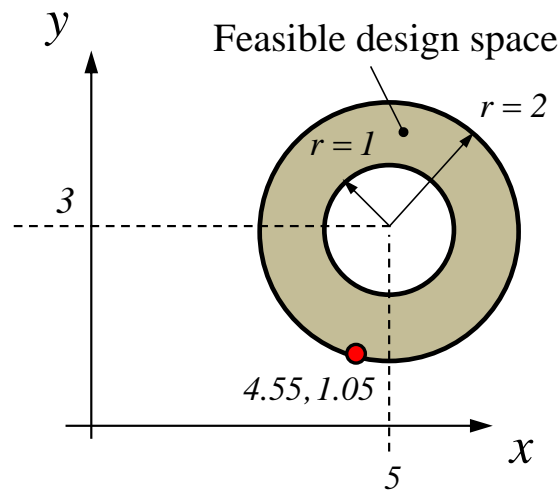


Figure 6.4 Feasible design space for Example 6.

The solution can be found by employing the standard minimization routine, *fmincon*, of Matlab. The following script is the main program that give an initial guess on the design variables, \underline{y}_0 , and set the minimization options. It lies outside the scope of these notes to go into details on the minimization options of the Optimization Toolbox of Matlab, however for most small- or medium-sized problems either

```
options=optimset('Algorithm','interior-point');
```

or

```
options=optimset('Algorithm','active-set');
```

should be adequate. In general, the 'interior-point' alternative will be most robust whereas the 'active-set' alternative will be faster. Next, the minimization routine is called. It needs four non-empty inputs: the reference to an m-script that computes the objective function (here: MAS407_Ex6_Obj.m) the initial guess, the reference to an m-script that computes the inequality constraints and the equality constraints (here: MAS407_Ex6_Con.m) and, finally, the minimization options. The full script look like this:

```
close all;
clear;
y0=[0 0];
options=optimset('Algorithm','active-set');
[x fval]=fmincon(@MAS407_Ex6_Obj,y0,[],[],[],[],[],[],@MAS407_Ex6_Con,options)
```

The sub-script for the objective function is: *MAS407_Ex6_Obj.m*:

```
function f=MAS407_Ex6_Obj(y)
x=y(1);
y=y(2);
f=x*y;
```

and the sub-script for the side constraints (there are no equality constraints, hence, it is returned empty) is *MAS407_Ex6_Con.m*:

```
function [g h]=MAS407_Ex6_Con(y)
x=y(1);
y=y(2);
r1=1;
r2=2;
x0=5;
y0=3;
g(1)=r1^2-(x-x0)^2-(y-y0)^2;
g(2)=(x-x0)^2+(y-y0)^2-r2^2;
h=[];
```

Executing the main program yields the following screen output:

```
Local minimum possible. Constraints satisfied.
```

```
fmincon stopped because the predicted change in the objective function
is less than the default value of the function tolerance and constraints
are satisfied to within the default value of the constraint tolerance.
```

```
<stopping criteria details>
```

```
Active inequalities (to within options.TolCon = 1e-006):
    lower      upper      ineqlin      ineqnonlin
               2
```

```
x =
```

```
    4.5497    1.0514
```

```
fval =  
4.7833
```

This minimum design point is indicated in Fig. 6.4.

Assume that an equality constraint is introduced, $x = 6.5$. In that case *MAS407_Ex6_Con.m* is simply augmented to:

```
function [g h]=MAS407_Ex6_Con(y)  
x=y(1);  
y=y(2);  
r1=1;  
r2=2;  
x0=5;  
y0=3;  
g(1)=r1^2-(x-x0)^2-(y-y0)^2;  
g(2)=(x-x0)^2+(y-y0)^2-r2^2;  
h(1)=x-6.5;
```

and executing the main script yields:

```
x =  
6.5000    1.6771  
  
fval =  
10.9013
```

CHAPTER 7: Control

In this chapter we will examine the speed and position control of a permanent magnet DC-motor. If we neglect the dynamics of the armature inductance and neglect external disturbances then we can set up the governing equations for the DC-motor in the laplace-domain graphically as illustrated in Fig. 7.1

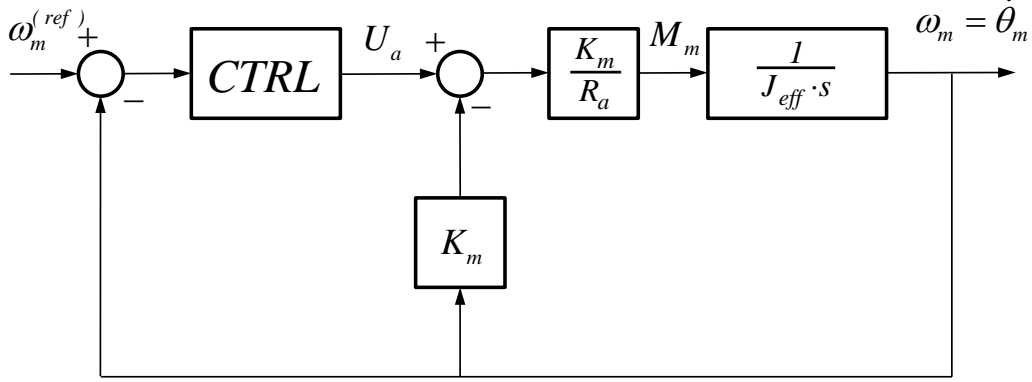


Figure 7.1 Speed control of a DC-motor.

In Fig. 7.1 it is assumed that the angular speed of the motor can be measured and that the armature voltage can be adjusted continuously. If the controller (CTRL) is taken to be a proportional controller with a gain K_p we get a first order system describing the relationship between the motor speed and the reference speed:

$$U_a = K_p \cdot (\omega_m^{(ref)} - \omega_m) \quad (7.1)$$

$$\frac{\omega_m}{\omega_m^{(ref)}} = \frac{K_{sys}}{\tau_{sys} \cdot s + 1} \quad (7.2)$$

$$K_{sys} = \frac{K_p}{K_p + K_m} \quad (7.3)$$

$$\tau_{sys} = \frac{R_a \cdot J_{eff}}{K_m \cdot (K_p + K_m)} \quad (7.4)$$

This means that a certain steady state error must be accepted, $K_{sys} < 1$. The steady error is reduced as K_p is increased, however, the gain cannot be increased indefinitely since voltage or current saturation will occur. Alternatively, the controller could be chosen as a proportional-integral controller with a gain K_p and a time constant τ_i . In that case we get a transfer function very similar to a second order system describing the relationship between the motor speed and the reference speed:

$$U_a = K_p \cdot \left(1 + \frac{1}{\tau_i \cdot s} \right) \cdot (\omega_m^{(ref)} - \omega_m) \quad (7.5)$$

$$\frac{\omega_m}{\omega_m^{(ref)}} = \frac{\tau_i \cdot s + 1}{\frac{s^2}{\omega_{sys}^2} + \frac{2 \cdot \zeta \cdot s}{\omega_{sys}} + 1} \quad (7.6)$$

$$\omega_{sys} = \sqrt{\frac{K_m \cdot K_p}{R_a \cdot J_{eff} \cdot \tau_i}} \quad (7.7)$$

$$\zeta_{sys} = \frac{1}{2} \cdot \sqrt{\frac{K_m}{K_p}} \cdot \sqrt{\frac{\tau_i}{R_a \cdot J_{eff}}} \cdot (K_p + K_m) \quad (7.8)$$

In order to behave as a similar as possible to a second order system the break frequency of the numerator should lie well above the natural eigenfrequency of the system. Introducing a ratio:

$$\lambda = \frac{\omega_i}{\omega_{sys}} = \frac{1}{\tau_i \cdot \omega_{sys}} \quad (7.9)$$

we will get close to second order system behavior if we ensure that $\lambda \geq 10$, i.e., the design of a suitable controller can be reduced to choosing λ and the desired system damping ζ_{sys} . In that case the controller parameters can be computed from:

$$K_p = \frac{K_m}{2 \cdot \lambda \cdot \zeta_{sys} - 1} \quad (7.10)$$

$$\tau_i = \frac{1}{\lambda^2} \cdot \frac{R_a \cdot J_{eff}}{K_m \cdot K_p} \quad (7.11)$$

For position control we have the setup illustrated in Fig. 7.2

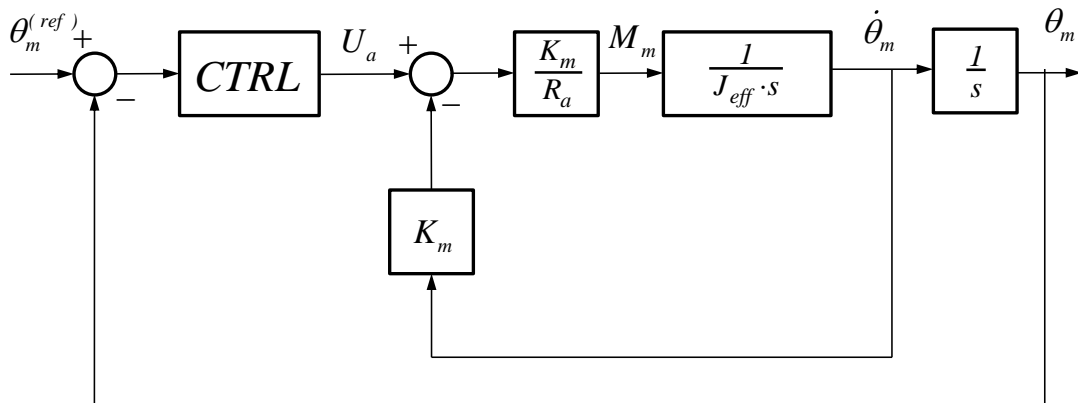


Figure 7.2 Position control of a DC-motor.

In Fig. 7.2 it is assumed that the angular position (rotation) of the motor can be measured and that the armature voltage can be adjusted continuously.

If the controller (CTRL) is taken to be a proportional controller with a gain K_p we get a second order system describing the relationship between the motor position and the reference position:

$$U_a = K_p \cdot (\theta_m^{(ref)} - \theta_m) \quad (7.12)$$

$$\frac{\theta_m}{\theta_m^{(ref)}} = \frac{I}{\frac{s^2}{\omega_{sys}^2} + \frac{2 \cdot \zeta \cdot s}{\omega_{sys}} + 1} \quad (7.13)$$

$$\omega_{sys} = \sqrt{\frac{K_m \cdot K_p}{R_a \cdot J_{eff}}} \quad (7.14)$$

$$\zeta_{sys} = \frac{I}{2} \cdot \sqrt{\frac{K_m^3}{K_p \cdot R_a \cdot J_{eff}}} \quad (7.15)$$

The gain can be determined from a desired system damping using (7.15):

$$K_p = \frac{I}{4} \cdot \frac{K_m^3}{R_a \cdot J_{eff} \cdot \zeta_{sys}^2} \quad (7.16)$$

CHAPTER 8: Multibody Dynamics

In this chapter we will briefly address some of the main modeling challenges associated with 3d-modeling of mechanical systems. The governing dynamic equations for a three-dimensional (spatial) body are:

$$\begin{aligned} m \cdot \ddot{\underline{r}}_G &= \underline{\Sigma F} \\ \underline{J}_{\underline{\underline{G}}} \cdot \underline{\dot{\omega}} + \underline{\tilde{\omega}} \cdot \underline{J}_{\underline{\underline{G}}} \cdot \underline{\omega} &= \underline{\Sigma M}_G \end{aligned} \quad (8.1)$$

In (8.1) the equations are written in matrix form as algebraic vectors and matrices. As in (3.1) m is the mass and $\underline{J}_{\underline{\underline{G}}}$ is the inertia matrix with respect to the mass center of the body. The angular velocity and angular acceleration are denoted, $\underline{\omega}$ and $\underline{\dot{\omega}}$, respectively. On the right hand side we have the sum of forces, $\underline{\Sigma F}$, and sum of moments around the mass center, $\underline{\Sigma M}_G$, acting on the body. The cross product is introduced via the skew symmetric matrix:

$$\underline{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \underline{\tilde{\omega}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (8.2)$$

Clearly, the governing equations are more complicated for spatial problems as compared to planar problems, however, the main challenge lies in keeping track of the degrees of freedom of the mechanical system and, especially, avoiding over-constraining and under-constraining. Any closed loop chain should have a minimum of zero-degrees of freedom (DOF), otherwise, it is over-constrained and if it is supposed to be able to move, then the DOF must be larger than zero. Similarly, for any closed loop chain each DOF represents two state variables (position and velocity), hence their initial values should be imposed on the model and the variation in time of each of the state variables should be controlled in one way or another. Otherwise, the model is under-constrained.

The DOF of a spatial mechanism is computed as:

$$DOF = 6 \cdot n_b - 6 \cdot n_f - 5 \cdot n_r - 5 \cdot n_p - 3 \cdot n_s \quad (8.3)$$

In (8.3) n_b is the number of bodies, n_f is the number of fixed connections, n_r is the number of revolute joints, n_p is the number of prismatic (or translational) joints, and n_s is the number of spherical joints. These types of joints are the most common but there exist several other different types of joints. An absolutely general constraint may lock any combination of the six-DOF's of a joint (three translations and three rotations).