Appendix

A.1 Normalization—The Important Step Towards Preparation for Programming

So far, the algorithms with their control variables and parameters were given in the originally derived physical form. An implementation or programming in this form is mainly impossible. Before this practical implementation can start, all algorithms have to be normalized and scaled if necessary, e.g. for fixed point and partly also for floating point processors. The purpose of the *normalization* consists of transferring these variables and parameters into a unity-less form and thus to prepare them for programming. The *scaling* is primarily necessary to increase the numerical accuracy which is of great importance for the use of fixed point processors.

This important step towards preparation for programming is demonstrated on two examples.

(a) *Example* 1:

The first Eq. of (3.55), which can be written in detail as follows, serves as the first example:

$$\begin{cases} i_{sd}(k+1) = \Phi_{11}\, i_{sd}(k) + \Phi_{12}\, i_{sq}(k) + \Phi_{13}\, \psi'_{rd}(k) + h_{11}\, u_{sd}(k) \\ i_{sq}(k+1) = -\Phi_{12}\, i_{sd}(k) + \Phi_{11}\, i_{sq}(k) - \Phi_{14}\, \psi'_{rd}(k) + h_{11}\, u_{sq}(k) \end{cases} \tag{A.1}$$

From (A.1) the following can be noticed:

- The variables like currents i_{sd}, i_{sq}, ψ'_{rd} and voltages u_{sd}, u_{sq} have to be normalized.
- From the parameters only Φ_{11}, Φ_{13} are already unity-less. All others have to be normalized.

N.P. Quang and J.-A. Dittrich, *Vector Control of Three-Phase AC Machines*,
Power Systems, DOI 10.1007/978-3-662-46915-6

For the normalization of the currents, the maximum inverter current $I_{\max}$ is often chosen. For the normalization of the voltage, the maximum value, which is $2U_{DC}/3$[1], is chosen. The quantity U_{DC} is itself variable and, with respect to the hardware, has to be normalized by $U_{\max}$ while measuring. The Eq. (A.1) is totally identical with the following:

$$\begin{cases} \dfrac{i_{sd}(k+1)}{I_{\max}} = \Phi_{11}\dfrac{i_{sd}(k)}{I_{\max}} + \Phi_{12}\dfrac{i_{sq}(k)}{I_{\max}} + \Phi_{13}\dfrac{\psi'_{rd}(k)}{I_{\max}} \\ \qquad\qquad + h_{11}\dfrac{2U_{\max}}{3I_{\max}}\left(\dfrac{U_{DC}(k)}{U_{\max}}\right)\left(\dfrac{u_{sd}(k)}{\frac{2}{3}U_{DC}}\right) \\ \dfrac{i_{sq}(k+1)}{I_{\max}} = -\Phi_{12}\dfrac{i_{sd}(k)}{I_{\max}} + \Phi_{11}\dfrac{i_{sq}(k)}{I_{\max}} - \Phi_{14}\dfrac{\psi'_{rd}(k)}{I_{\max}} \\ \qquad\qquad + h_{11}\dfrac{2U_{\max}}{3I_{\max}}\left(\dfrac{U_{DC}(k)}{U_{\max}}\right)\left(\dfrac{u_{sq}(k)}{\frac{2}{3}U_{DC}}\right) \end{cases} \tag{A.2}$$

In the Eq. (A.2) new symbols are introduced and replaced:

$$\begin{aligned} i^N_{sd,sq} &= \frac{i_{sd,sq}}{I_{\max}};\ \psi'^N_{rd} = \frac{\psi'_{rd}}{I_{\max}};\ u^N_{sd,sq} = \frac{u_{sd,sq}}{2U_{DC}/3} \\ h^N_{11} &= k_u\, U^N_{DC};\ k_u = h_{11}\frac{2U_{\max}}{3I_{\max}};\ U^N_{DC} = \frac{U_{DC}}{U_{\max}} \end{aligned} \tag{A.3}$$

Superscripts N: normalized quantities

The parameters Φ_{12}, Φ_{14} are frequency dependent. The value $f_{\max}$ is used for the normalization of the frequencies. Using (3.54) it can be written:

$$\begin{aligned} \Phi_{12} &= \omega_s T = 2\pi f_s T = (2\pi f_{\max} T)\left(\frac{f_s}{f_{\max}}\right) = k_{f1} f^N_s \\ \Phi_{14} &= \frac{1-\sigma}{\sigma}\omega T = \frac{1-\sigma}{\sigma}2\pi f T = \left(\frac{1-\sigma}{\sigma}2\pi f_{\max} T\right)\left(\frac{f}{f_{\max}}\right) \\ &= k_{f2} f^N \end{aligned} \tag{A.4}$$

In (A.4) the symbols mean:

$$f^N_s = \frac{f_s}{f_{\max}};\ f^N = \frac{f}{f_{\max}};\ k_{f1} = 2\pi f_{\max} T;\ k_{f2} = \frac{1-\sigma}{\sigma}2\pi f_{\max} T \tag{A.5}$$

[1] U_{DC}: DC link voltage of the inverter.

The Eq. (A.1) can now be rewritten in the normalized form, in which the constants k_u, k_{f1} and k_{f2} as well as the constant parameters Φ_{11}, Φ_{13} have to be calculated only at the beginning, i.e. at the initialization of the system.

$$\begin{cases} i_{sd}^N(k+1) = \Phi_{11}\, i_{sd}^N(k) + \Phi_{12}\, i_{sq}^N(k) + \Phi_{13}\, \psi_{rd}^{/N}(k) \\ \qquad\qquad + h_{11}^N\, u_{sd}^N(k) \\ i_{sq}^N(k+1) = -\Phi_{12}\, i_{sd}^N(k) + \Phi_{11}\, i_{sq}^N(k) - \Phi_{14}\, \psi_{rd}^{/N}(k) \\ \qquad\qquad + h_{11}^N\, u_{sq}^N(k) \end{cases} \tag{A.6}$$

The original Eq. (A.1) exists now in programmable form without loss of its physical meaning. *The voltages in this normalized form represent the degree of modulation* where the variable DC link voltage U_{DC} is considered by the parameter h_{11}^N, which has to be updated on-line.

To achieve the most possible numerical accuracy with fixed point or integer arithmetic, the normalized quantities (represented in hexadecimal form) are shifted to the left (multiplied by 2) as much as possible without producing overflow. For normalized currents, voltages and frequencies which assume only values smaller than one, the multiplication factor can be e.g. 2^{15} for 16 bit fixed point processors. This process is commonly described as *scaling*. The multiplication factor of 2^{15} is the *scaling factor* which at the same time means the number of digits behind the comma. For parameters, which by their nature are already greater than one, the scaling has to be carried out in a way that on one hand the maximum word length is utilized, but on the other hand overflow is avoided simultaneously. In principle, this problem does not exist any-more with the use of floating point processors and only appears for conversions between data types again, e.g. between integer numbers and signed floating point numbers.

(b) *Example* 2:

A further typical example is shown by normalizing and scaling of the quantities within the Eq. (A.6) for the calculation of the angular velocity ω.

$$\begin{aligned} \vartheta(k+1) &= \vartheta(k) + \omega(k)T \\ &= \vartheta(k) + 2\pi f(k)T \end{aligned} \tag{A.7}$$

If the values 2π and f_{max} are chosen as normalizing quantities for the angle and frequency, and it is considered that:

- the frequency has to be signed (e.g. positive for right and negative for left rotation), and
- the angle has to be unsigned (i.e. only forwards counting 0, π, 2π, 3π, 4π…)

then it is for example possible to use the number 2^{15} as scaling factor for the frequency at 16 bit word length, and 2^{16} for the angle. From (A.7) the following will be obtained:

$$\left[2^{16}\frac{\vartheta(k+1)}{2\pi}\right] = \left[2^{16}\frac{\vartheta(k)}{2\pi}\right] + \left[2^{15}\frac{f}{f_{\max}}\right](2f_{\max}T) \tag{A.8a}$$

or

$$\left[2^{16}\,\vartheta^N(k+1)\right] = \left[2^{16}\,\vartheta^N(k)\right] + \left[2^{15}f^N(k)\right]k_{f3} \tag{A.8b}$$

The Eq. (A.8b) means:

- $2^{16} \times \vartheta^N$ the unsigned integer calculation quantity for the angle,
- $2^{15} \times f^N$ the signed calculation quantity for the frequency

A.2 Example for the Model Discretization in the Sect. 3.1.2

A system of second order following (3.5) is given by:

$$\mathbf{A} = \begin{bmatrix} a & \omega \\ -\omega & a \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.9}$$

(a) *Method* 1: Series expansion with truncation after the linear term (Euler). The use of (3.14) provides:

$$\mathbf{\Phi} = \begin{bmatrix} 1+aT & \omega T \\ -\omega T & 1+aT \end{bmatrix}; \quad \mathbf{H} = T\begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} \tag{A.10}$$

(b) *Method* 2: Series expansion with truncation after the quadratic term. The use of (3.14) provides again:

$$\mathbf{\Phi} = \begin{bmatrix} 1+aT+\left[(aT)^2-(\omega T)^2\right]\big/2 & \omega T(1+aT) \\ -\omega T(1+aT) & 1+aT+\left[(aT)^2-(\omega T)^2\right]\big/2 \end{bmatrix} \tag{A.11a}$$

$$\mathbf{H} = bT\begin{bmatrix} 1+aT/2 & \omega T/2 \\ -\omega T/2 & 1+aT/2 \end{bmatrix} \tag{A.11b}$$

(c) *Method* 3: Euler discretization in a suitable coordinate system.

This method is applicable if the system matrix **A** owns the symmetry properties of the special block diagonal structure (A.9) which is often the case for the modeling of three-phase machines. In this case, state and input quantities can be understood as complex variables.

$$\begin{aligned} \mathbf{x} &= x_x + j\,x_y \\ \dot{\mathbf{x}}(t) &= (a - j\omega)\,\mathbf{x}(t) + b\,\mathbf{u}(t) \end{aligned} \tag{A.12}$$

At first, the continuous system is viewed in a coordinate system which circulates with the frequency $-\omega$ with respect to the target coordinate system, i.e. $\mathbf{x} = \mathbf{x}^{\omega} e^{-j\omega t}$ (regarding the topic "transformation of coordinates" cf. Chap. 1). Considering the product rule, the following is obtained for the time derivative:

$$\dot{\mathbf{x}}^{\omega}(t) = a\,\mathbf{x}^{\omega}(t) + b\,\mathbf{u}^{\omega}(t) \tag{A.13}$$

The discretization using Euler method leads to the following discrete state equation:

$$\mathbf{x}^{\omega}(k+1) = (1 + aT)\,\mathbf{x}^{\omega}(k) + bT\,\mathbf{u}^{\omega}(k) \tag{A.14}$$

For the inverse transformation into the target coordinate system, the Eq. (A.14) has to be subjected to the counter-rotation, i.e.

$$\mathbf{x}^{\omega}(k) = \mathbf{x}(k) e^{j\vartheta(k)},\ \mathbf{x}^{\omega}(k+1) = \mathbf{x}(k+1) e^{j\vartheta(k+1)} \tag{A.15}$$

Thereat, the discrete transformation angle ϑ is expressed by Euler discretization as follows:

$$\vartheta(k+1) = \vartheta(k) + \omega\,T \tag{A.16}$$

The state equation is now in the target coordinate system:

$$\mathbf{x}(k+1) = e^{-j\omega T}\left[(1 + aT)\,\mathbf{x}(k) + bT\,\mathbf{u}(k)\right] \tag{A.17}$$

or resolved with discrete transfer matrices:

$$\mathbf{\Phi} = (1 + aT)\begin{bmatrix} \cos\omega T & \sin\omega T \\ -\sin\omega T & \cos\omega T \end{bmatrix};\ \mathbf{H} = bT\begin{bmatrix} \cos\omega T & \sin\omega T \\ -\sin\omega T & \cos\omega T \end{bmatrix} \tag{A.18}$$

(d) *Method* 4: Substitute function using the Sylvester-Lagrange substitute polynomials.

The eigen values of the continuous system matrix $\mathbf{A}$ will be:

$$\lambda_{1,2} = a \pm j\omega$$

It follows then for Eqs. (3.19)–(3.22):

$$\begin{aligned} M(\lambda) &= (\lambda - \lambda_1)(\lambda - \lambda_2) \\ M_1 &= (\lambda - \lambda_1)\,, \quad M_2 = (\lambda - \lambda_2) \\ m_1 &= (\lambda_1 - \lambda_2)\,, \quad m_2 = (\lambda_2 - \lambda_1) \\ R(\lambda) &= \frac{\lambda_1\, e^{\lambda_2 T} - \lambda_2\, e^{\lambda_1 T}}{\lambda_1 - \lambda_2} + \lambda \frac{e^{\lambda_1 T} - e^{\lambda_2 T}}{\lambda_1 - \lambda_2} \end{aligned} \tag{A.19}$$

Therewith the substitute function $\mathbf{R}(\mathbf{A})$ can be given as:

$$\mathbf{R}(\mathbf{A}) = \mathbf{\Phi} = \frac{\lambda_1\, e^{\lambda_2 T} - \lambda_2\, e^{\lambda_1 T}}{\lambda_1 - \lambda_2}\,\mathbf{I} + \frac{e^{\lambda_1 T} - e^{\lambda_2 T}}{\lambda_1 - \lambda_2}\mathbf{A} \tag{A.20}$$

and finally the system matrix $\mathbf{\Phi}$ becomes:

$$\mathbf{\Phi} = e^{aT} \begin{bmatrix} \cos \omega T & \sin \omega T \\ -\sin \omega T & \cos \omega T \end{bmatrix} \tag{A.21}$$

The input matrix $\mathbf{H}$ is calculated by direct integration of $\mathbf{\Phi}$ according to (3.12):

$$\mathbf{H} = \frac{b}{a^2 + \omega^2} \begin{bmatrix} e^{aT}(a\ \cos \omega T + \omega\ \sin \omega T) - a & e^{aT}(a\ \sin \omega T - \omega\ \cos \omega T) + \omega \\ -e^{aT}(a\ \sin \omega T - \omega\ \cos \omega T) + \omega & e^{aT}(a\ \cos \omega T + \omega\ \sin \omega T) - a \end{bmatrix} \tag{A.22}$$

A.3 Application of the Method of the Least Squares Regression

The method of the least squares regression is often used for the optimization of control loops or the identification of the system parameters. The goal is normally to find an approximate function $y(x)$ in the form of a polynomial of nth order

$$y(x) = a_0 + a_1\, x + a_2\, x^2 + \cdots + a_n\, x^n \tag{A.23}$$

from a set of m experimental measurement pairs $[y_i, x_i]$, (i = 1, 2, 3, …, m) and by the prerequisite, that the loss function (cf. Rojiani 1996):

$$S = \sum_{i=1}^{m} [y_i - y(x_i)]^2 \tag{A.24}$$

is minimized. A typical application example is the off-line identification of the main inductance L_s (cf. Sect. 6.4.4, Fig. 6.18) in dependence on the magnetizing current i_μ. As an approach for $L(i)$[2] a polynomial of 4th order, thus n = 4, is very suitable.

$$L(i) = a_0 + a_1\, i + a_2\, i^2 + a_3\, i^3 + a_4\, i^4 \tag{A.25}$$

The task is now to determine the coefficients a_0, a_1, a_2, a_3 and a_4 from m pairs $[L_i, i_i]$ with (i = 1, 2, 3, …, m). To minimize the loss function, at first (A.25) has to be inserted into (A.24), and then the partial derivations

$$\frac{\partial S}{\partial a_0};\ \frac{\partial S}{\partial a_1};\frac{\partial S}{\partial a_2};\frac{\partial S}{\partial a_3};\ \frac{\partial S}{\partial a_4} \tag{A.26}$$

have to be set to zero. Thereby a system with (n + 1) = 5 linear equations results (cf. Rojiani 1996):

$$\begin{bmatrix} m & \sum_{i=1}^{m} i_i & \sum_{i=1}^{m} i_i^2 & \sum_{i=1}^{m} i_i^3 & \sum_{i=1}^{m} i_i^4 \\ \sum_{i=1}^{m} i_i & \sum_{i=1}^{m} i_i^2 & \sum_{i=1}^{m} i_i^3 & \sum_{i=1}^{m} i_i^4 & \sum_{i=1}^{m} i_i^5 \\ \sum_{i=1}^{m} i_i^2 & \sum_{i=1}^{m} i_i^3 & \sum_{i=1}^{m} i_i^4 & \sum_{i=1}^{m} i_i^5 & \sum_{i=1}^{m} i_i^6 \\ \sum_{i=1}^{m} i_i^3 & \sum_{i=1}^{m} i_i^4 & \sum_{i=1}^{m} i_i^5 & \sum_{i=1}^{m} i_i^6 & \sum_{i=1}^{m} i_i^7 \\ \sum_{i=1}^{m} i_i^4 & \sum_{i=1}^{m} i_i^5 & \sum_{i=1}^{m} i_i^6 & \sum_{i=1}^{m} i_i^7 & \sum_{i=1}^{m} i_i^8 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} L_i \\ \sum_{i=1}^{m} i_i\, L_i \\ \sum_{i=1}^{m} i_i^2\, L_i \\ \sum_{i=1}^{m} i_i^3\, L_i \\ \sum_{i=1}^{m} i_i^4\, L_i \end{bmatrix} \tag{A.27}$$

The system (A.27) can be merged into the following form:

$$\mathbf{A}[5,\ 5]\ *\ \mathbf{a}[5] = \mathbf{b}[5] \tag{A.28}$$

thereat $\mathbf{A}[5,\ 5]$ and $\mathbf{b}[5]$ are given by the measurement pairs, following (A.27). If C is used as programming language, then the calculation can be realized by the following program section as an example, where $\mathbf{A}[5,\ 5]$ and $\mathbf{b}[5]$ are summarized in a matrix $\mathbf{A}[5,\ 6]$ with $\mathbf{b}[5]$ as the 6th column. In this example it is assumed, that "MeasNum" is the number of the measurement pairs and "PolyOrder" the order of the approached polynomial. Here it holds:

MeasNum = 10; PolyOrder = 4

[2]For simplification L_s is replaced by L, and i_μ by i.

During the measurement the current is increased by 0,1 × ImNominal step by step from 0,1 × ImNominal to ImNominal (nominal magnetizing current). The 10 measured inductance values are stored in the field *L*[0]…L[9].

```
/* Calculation of the sums or the elements of the matrix A[ ][ ] */
s[0] = (float)MeasNum;
for (i = 1; i <= 2*PolyOrder; i++)
   {  s[i] = 0.0;
      for (j = 1; j <= MeasNum; j++)
         s[i] += pow(ImNominal * (float)j/10.0, i);
   }
/* The sums are assigned to the elements of the matrix A[ ][ ] */
for (i = 0; i <= PolyOrder; i++)
   for (j = 0; j <= PolyOrder; j++)
      A[i][j] = s[i][j];
/* Calculation of the vector b[ ] as the 6th column of the matrix A[ ][ ] */
A[0][PolyOrder+1] = 0.0;
for (i = 0; i < MeasNum; i++)
   A[0][PolyOrder+1] += L[i];
for (i = 1; i <= PolyOrder; i++)
   {  A[i][PolyOrder+1] = 0.0;
      for (j = 0; j < MeasNum; j++)
         A[i][PolyOrder+1] += L[j] * pow(ImNominal * (float)(j+1)/10.0, i);
   }
```

After calculation of $\mathbf{A}[5, 5]$ and $\mathbf{b}[5]$ the linear equation system (A.27) or (A.28) can be relatively simply solved by using the *Gauss elimination method*. The first step of the method is the *forward elimination*, which can be summarized (cf. Sedgewick 1992) as follows:

The first variable in all equations, with exception of the first one, has to be eliminated by addition of suitable multiples of the first equation to each of the other equations. Then the second variable in all equations, with exception of the first two, has to be eliminated by addition of suitable multiples of the second equation to each of the equations from the third up to the last one (now named as the *N*-th). Then the third variable in all equations, with the exception of the first three, has to be eliminated etc. To eliminate the *i*-th variable in the *j*-th equation (for *j* between $i + 1$ and *N*), the *i*-th equation must be multiplied with a_{ji}/a_{ii} and subtracted from the *j*-th equation.

The described procedure however is too simple to be completely right: a_{ii} (now named as *pivot element*) can become zero, so that a division by zero could arise. This can be avoided because any arbitrary row (from the $(i + 1)$-th to the *N*-th) can be swapped with the i-th row, so that a_{ii} in the outer loop is different from zero. For swapping it is the best approach to use the row for which the value in the *i*-th

column is the greatest with respect to the absolute amount. The reason is, that in the calculation considerable errors can arise if the pivot value, which is used to multiply a row by a factor, is very small. If a_{ii} is very small, a_{ji}/a_{ii} can become very big. This process, called the partial pivoting, is realized in the example of the L_s identification by the following program section.

```
/* Forward elimination: partial pivoting */
for (i = 0; i <= PolyOrder; i++)
  {  max = i;
     for (j = i+1; j <= PolyOrder; j++)
        if (fabs(A[j][i]) > fabs(A[max][i])) max = j;
     for (k = i; k <= PolyOrder+1; k++)
        {  Temp = A[i][k];                    /* Temp: temporary variable */
                  A[i][k] = A[max][k];
                           A[max][k] = Temp;  }
     for (j = i+1; j <= PolyOrder; j++)
        for (k = PolyOrder+1; k >= i; k– –)
           A[j][k] –= A[i][k] * A[j][i] / A[i][i]
  }
```

After the forward elimination step is completed, the field below the diagonal of the modified matrix **A**[5][5] contains only zeros. The step of the *backwards insertion* can be executed now to calculate the coefficients a_0, a_1, a_2, a_3 and a_4.

```
/* Backwards insertion: Calculation of a0, a1, a2, a3, a4,
   then storage in a[0]...[4] */
for (j = PolyOrder; j >= 0; j– –)
  {  Temp = 0.0;
     for (k = j+1; k <= PolyOrder; k++) Temp += A[j][k] * a[k];
     a[j] = (A[j][PolyOrder+1] – Temp) / A[j][j];
  }
```

With the calculated coefficients a_0, a_1, a_2, a_3 and a_4, the magnetizing curve $L(i)$ in the form of a polynomial (cf. Eq. (A.25)) is now available.

A.4 Definition and Calculation of Lie Derivation

An unexcited system (input vector $\mathbf{u} = \mathbf{0}$) is defined (cf. Wey 2001, Appendix B) as follows:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) \tag{A.29}$$

A scalar function $v(\mathbf{x})$ is given. The derivation of this scalar function along the freely moving state trajectory (along the vector field $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n$) of the unexcited system (A.29):

$$\mathbf{x}(t) = \mathbf{\Phi}_t^f(\mathbf{x}_0) \tag{A.30}$$

can be given as follows:

$$L_f v(\mathbf{x}) = \sum_{i=1}^{n} \left[\frac{\partial v(\mathbf{x})}{\partial x_i} f_i(x_1, \cdots, x_n) \right] \tag{A.31}$$

with:

$$\frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial v}{\partial x_1}, \quad \frac{\partial v}{\partial x_2}, \quad \cdots, \quad \frac{\partial v}{\partial x_n} \right] \tag{A.32}$$

Using (A.32), the Lie derivation $L_f v(\mathbf{x})$ can also be formulated as a scalar product (a scalar function):

$$L_f v = \frac{\partial v}{\partial \mathbf{x}} \mathbf{f} \quad \Rightarrow \quad L_f v(\mathbf{x}) = \frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) \tag{A.33}$$

The function $L_f v(\mathbf{x})$ returns the quantitative change of $v(\mathbf{x})$ along the trajectory (A.30). Figure A.1 illustrates this.

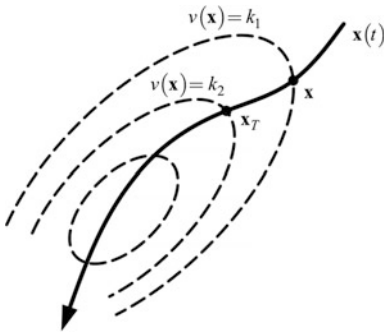


Fig. A.1 Derivation of the scalar function $v(\mathbf{x})$ along the state trajectory $\mathbf{x}(t)$

The dashed curves in the Fig. A.1 represent the sets of points inside $\mathbb{R}^n$ at which the function $v(\mathbf{x})$ has the same values. The dashed curve with the point $\mathbf{x}$ contains the set of points which fulfills $v(\mathbf{x}) = k_1$, and the curve with $\mathbf{x}_T$ the set of points fulfilling $v(\mathbf{x}_T) = k_2$. In this case, the speed of the quantitative change of $v(\mathbf{x})$ along $\mathbf{x}(t)$, from point $\mathbf{x}$ to the point $\mathbf{x}_T$, will be:

$$\begin{aligned} L_f v(\mathbf{x}) &= \lim_{T\to 0} \frac{k_2 - k_1}{T} = \lim_{T\to 0} \frac{v(\mathbf{x}_T) - v(\mathbf{x})}{T} = \lim_{T\to 0} \frac{v\left(\mathbf{\Phi}_T^f(\mathbf{x})\right) - v(\mathbf{x})}{T} \\ &= \frac{\partial v}{\partial \mathbf{x}} \lim_{T\to 0} \frac{\mathbf{\Phi}_T^f(\mathbf{x}) - \mathbf{x}}{T} = \frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} = \frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) \end{aligned} \tag{A.34}$$

The Lie derivation has the following properties:

- The multiple Lie derivation of $v(\mathbf{x})$, at first along the vector field $\mathbf{f}(\mathbf{x})$ and then along $\mathbf{g}(\mathbf{x})$, can be written as follows:

$$L_g L_f v(\mathbf{x}) = L_g \left[L_f v(\mathbf{x})\right] = \frac{\partial \left[L_f v(\mathbf{x})\right]}{\partial \mathbf{x}} \mathbf{g}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \left[\frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})\right] \mathbf{g}(\mathbf{x}) \tag{A.35}$$

- Let $w(\mathbf{x})$ be an additional scalar function, then the following relation is valid:

$$\begin{aligned} &L_{wf} v(\mathbf{x}) = \left[L_f v(\mathbf{x})\right] w \\ &\text{because} \quad L_{wf} v(\mathbf{x}) = \frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} (w\,\mathbf{f}) = \underbrace{\left[\frac{\partial v(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})\right]}_{L_f v(\mathbf{x})} w(\mathbf{x}) \end{aligned} \tag{A.36}$$

- Let k be an integer number, then the k-fold Lie derivation of $v(\mathbf{x})$ along $\mathbf{f}(\mathbf{x})$ can be recursively calculated as follows:

$$L_f^k v(\mathbf{x}) = \frac{\partial \left[L_f^{k-1} v(\mathbf{x})\right]}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}) \quad \text{with} \quad L_f^0 v(\mathbf{x}) = v(\mathbf{x}) \tag{A.37}$$

A.5 Example for Rest-to-Rest Trajectory of Differentially Flat Systems

The DC motor (Fig. A.2) is chosen as an example to explain the design of a set point trajectory for flat systems. The motor is described by the following equation system.

- The armature voltage:

$$u_A = e_A + R_A i_A + L_A \frac{di_A}{dt} \tag{A.38}$$

- The induced electromotive force:

$$e_A = k_e \psi_M n \tag{A.39}$$

- The motion equation:

$$m_M = m_L + 2\pi J \frac{dn}{dt} \tag{A.40}$$

- The electric torque:

$$m_M = k_M \psi_M i_A \tag{A.41}$$

- The electric constant:

$$k_e = 2\pi k_M \tag{A.42}$$

- The armature time constant:

$$T_A = \frac{L_A}{R_A} \tag{A.43}$$

The motor has the following parameters (Fig. A.2):

Armature resistance:	R_A = 250 mΩ	*Inertia:*	J = 0,012 kgm^2
Armature inductance:	L_A = 4 mH	*Motor constants:*	k_e = 236,8
Nominal motor flux:	$\psi_M = 0,04$ Vs		k_M = 38,2

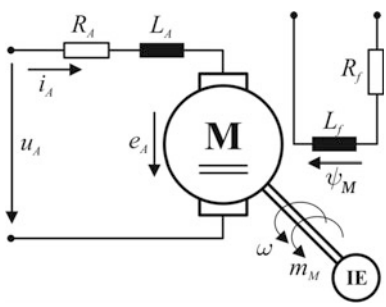


Fig. A.2 Electric circuit of separately excited DC motors

It is easy to see that the motor speed n presents the flat output. According to the first condition (3.116), the flat output must be a function of input and state variables and its derivatives. The output n is at the same time the state variable. The condition (3.116) is fulfilled.

Seeing the load torque m_L as an unknown disturbance and after some transformations using the Eqs. (A.38)–(A.43) we obtain:

$$i_A = \frac{m_M}{k_M \psi_M} = \frac{2\pi J \dot{n}}{k_M \psi_M} \tag{A.44}$$

After inserting (A.39) and (A.44) into (A.38), the armature voltage can be computed by:

$$u_A = k_e \psi_M n + R_A \frac{2\pi J \dot{n}}{k_M \psi_M} + L_A \frac{2\pi J \ddot{n}}{k_M \psi_M} \tag{A.45}$$

With (A.45) the second condition (3.117) is also fulfilled. The conclusion: the DC motor is differentially flat with the flat output n.

While moving from the starting speed n_0 to the end speed n_E, we can predict that the two points n_0, n_E are equilibrium points. That means the derivatives of the speed at these points must be zero. Because the voltage in (A.45) contains derivatives of n up to 2nd order, the trajectory for n^* has to be differentiable at least to 2nd order.

With $j = 1$ (only one output) and $r = 2$ (differentiable to 2nd order), the formula (3.146) is used, and the result is obtained as:

$$n^*(t) = n_E \left(\frac{t - t_0}{t_E - t_0}\right)^2 \left[3 - 2\left(\frac{t - t_0}{t_E - t_0}\right)\right] \tag{A.46}$$

However, for the purpose of reducing unwanted oscillations at starting, and simultaneously smoothing the trajectory, we can choose n^* to be differentiable to higher order. For example:

$$n^* = n_E \left(\frac{t - t_0}{t_E - t_0}\right)^3 \left[10 - 15\left(\frac{t - t_0}{t_E - t_0}\right) + 6\left(\frac{t - t_0}{t_E - t_0}\right)^2\right] \tag{A.47}$$

or:

$$n^* = n_E \left(\frac{t - t_0}{t_E - t_0}\right)^5 \left[126 - 420\left(\frac{t - t_0}{t_E - t_0}\right) + 540\left(\frac{t - t_0}{t_E - t_0}\right)^2 - 315\left(\frac{t - t_0}{t_E - t_0}\right)^3 + 70\left(\frac{t - t_0}{t_E - t_0}\right)^4\right] \tag{A.48}$$

In a flatness-based control structure of an externally excited DC motor, the feed forward component of the armature voltage will be computed using (A.45). Hereby, the reference speed n^* is calculated from one of the Eqs. (A.46), (A.47) or (A.48).

$$u_A^f = k_e \psi_M n^* + R_A \frac{2\pi J \dot{n}^*}{k_M \psi_M} + L_A \frac{2\pi J \ddot{n}^*}{k_M \psi_M} \tag{A.49}$$

Now we want to investigate the influence of the time $T = t_E - t_0$ on the system stability using simulation.

- Calculation of n^* using (A.46)
  The set point n^* is differentiable up to 2nd order. Figure A.3 shows that after halving the time T from 0.1 s to 0.05 s the ramp-up process becomes unstable.
- Calculation of n^* using (A.47)
  Because the use of (A.46) makes the ramp-up process unstable with T = 0.05 s, we change now to the formula (A.47), with the result: the ramp-up is stable (Fig. A.4, left). But, if we decrease T down to 0.03 s, the ramp-up becomes unstable again.
- Calculation of n^* using (A.48)
  An improvement for the case in Fig. A.4 (right) with $T = 0.03$ s will be achieved in a simple way: we switch to the formula (A.48). The stable result is shown in the Fig. A.5.
  We can now summarize that an increase of the order of differentiability of n^* always results in a more stable behavior and a smoother ramp-up procedure. At the same time, the start-up time T can be usefully reduced.

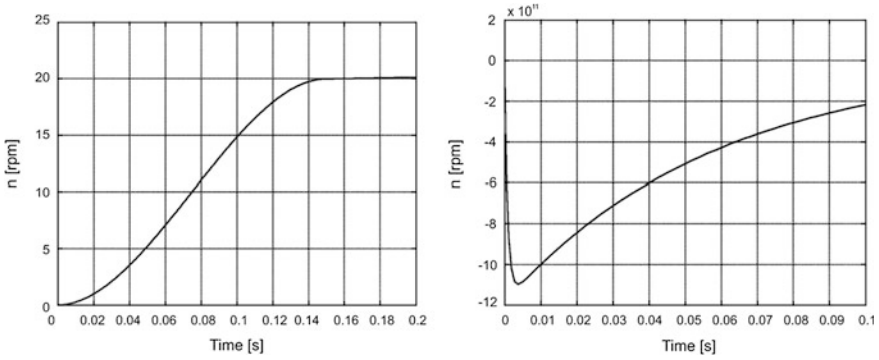


Fig. A.3 Speed ramp with $T = 0.1$ s (*left*) and $T = 0.05$ s (*right*)

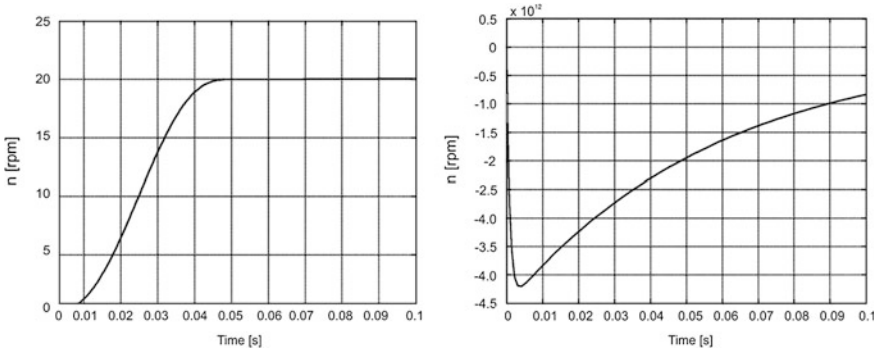
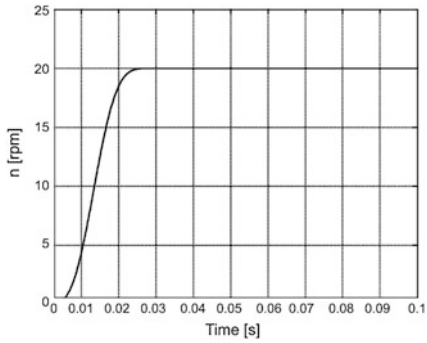


Fig. A.4 Speed ramp with $T = 0.05$ s (*left*) and $T = 0.03$ s (*right*)

Fig. A.5 Speed ramp with $T = 0.03$ s



References

Lévine J (2009) Analysis and control of nonlinear systems: a flatness-based approach. Springer Dordrecht Heidelberg London New York

Rojiani KB (1996) Programming in C with numerical methods for engineers. Prentice-Hall International, Inc.

Sedgewick R (1992) Algorithmen in C. Addison-Wesley

Wey T (2001) Nichtlineare Regelungssysteme: Ein differentialalgebraischer Ansatz. B.G. Teubner Stuttgart - Leipzig - Wiesbaden.

Index

N.P. Quang and J.-A. Dittrich, *Vector Control of Three-Phase AC Machines*,
Power Systems, DOI 10.1007/978-3-662-46915-6