

Circuit Simulation

Compendium

Helmut Graeb

Version 1.0 (SS 24) Helmut Graeb

The presentation follows: “Kurt Antreich: Circuit Simulation/Electronic Design Automation, Lecture Notes, Technische Universität München, ~1980 – 2003”. The following compendium represents a supplement to these notes, not a replacement.

No textbook matches this compendium. Related books are, e.g., the following:

- [1] Jan Ogrodzki: Circuit Simulation Methods and Algorithms, CRC Press, 2018.
- [2] Ibrahim N. Hajj: Computational Methods in Circuit Simulation, CreateSpace Independent Publishing Platform, 2016.
- [3] L. T. Pillage, R. A. Rohrer and C. Visweswariah: Electronic and System Simulation Methods, McGraw-Hill, Inc., 1995.
- [4] J. Vlach und K. Singhal: Computer Methods for Circuit Analysis and Design, Van Nostrand Reinhold, 2nd edition, 1994.
- [5] C. Ho, A. F. Ruehli, P. A. Brennan, The Modified Nodal Approach to Network Analysis, IEEE Trans. CAS, v. 25, pp. 504–509, June 1975.
- [6] L. W. Nagel, D. O. Pederson: SPICE (Simulation Program with Integrated Circuit Emphasis), Memorandum No. ERL-M382, University of California, Berkeley, April 1973.
- [7] D. A. Calahan: Computer-Aided Network Design, McGraw-Hill, 1972.

Status: 31 January 2024

Copyright 2024

Circuit Simulation

Compendium

Technical University of Munich

Chair of Electronic Design Automation

Arcisstr. 21

80333 Munich, Germany

All rights reserved.

Inhaltsverzeichnis

1.	Introduction.....	6
1.1	Nonlinear circuit parts.....	6
1.2	Differential circuit parts	7
1.3	In the following,	8
2.	Computer-based circuit representation and analysis	9
2.1	Nodal voltage system	11
2.1.1	Kirchhoff's current law (KCL) at all circuit nodes	11
2.1.2	Branch constitutive equations (BCE), i.e., Ohm's law in all branches.....	12
2.1.3	Kirchhoff's voltage law (KVL) for all branch voltages	12
2.2	Modified nodal voltage system/modified nodal analysis (MNA)	14
2.3	Integration of controlled sources into the nodal voltage system.....	18
2.4	Setup rules for the modified nodal system	19
2.5	Connecting subsystems.....	20
2.6	Advantages of the nodal voltage system	21
2.7	Modified nodal system by triangular decomposition of sparse network tableau.....	22
2.8	Source transformations.....	23
3.	AC analysis.....	27
3.1	Illustrating the Gaussian elimination (LU decomposition) with an example.....	27
3.2	Gaussian elimination (LU decomposition).....	29
3.3	Complexity of the Gaussian elimination	30
3.4	Parallel simulation approach	31
3.4.1	Example of the parallel simulation approach.....	32
3.5	Interpreting a Gaussian elimination step by circuitry	33
3.6	Structural interpretation of a Gaussian elimination step.....	35
3.7	Computational cost of a Gaussian elimination step considering zero-elements	36
3.8	Structural Gaussian elimination considering zero-elements.....	38
3.8.1	Suboptimal Pivoting Rules	38
3.8.2	Markovitz Pivoting	41
4.	DC analysis.....	43
4.1	Bi-sectioning (one-dimensional) for root finding.....	44

4.1.1	Convergence rate of bi-sectioning	45
4.2	Newton-Raphson method for root finding (one-dimensional)	45
4.2.1	Convergence rate of Newton-Raphson	46
	Single root: $F'(x^*) \neq 0$	47
	Root of order n : $F'(x^*) = \dots = F^{(n-1)}(x^*) = 0$ $F^{(n)}(x^*) \neq 0$	48
4.3	Convergence rate of an arbitrary iteration function for root finding (one-dimensional)	48
4.4	Multivariate Newton-Raphson iteration function for root finding	50
4.4.1	Convergence rate of Newton-Raphson	51
4.5	Termination criteria for the iterative root finding	52
5.	The linearized nodal voltage system in iteration step $\mu+1$	53
5.1	Linearizing the nonlinear nodal system equations	53
5.2	Linearizing the branch constitutive equations	55
5.3	Equivalent circuit model for linearized branch constitutive equations	57
5.4	Modified Newton-Raphson method	59
6.	TR analysis	61
6.1	Determining the continuous step response of a simple linear circuit	61
6.2	Determining the discrete step response of a simple linear circuit	62
6.3	First glance at error and stability of numerical integration	64
6.4	Mathematical task of TR analysis	65
6.5	Four properties of numerical integration methods	66
6.6	The test differential equation	66
6.7	The difference equations of four simple numerical integration methods	66
6.7.1	Forward Euler method's difference equation	67
6.7.2	Backward Euler method's difference equation	68
6.7.3	Trapezoidal method's difference equation	68
6.7.4	Second-order Gear method's difference equation	69
6.8	Properties of the four simple numerical integration methods	70
6.8.1	Forward Euler method's properties	70
6.8.2	Backward Euler method's properties	74
6.8.3	Trapezoidal method's properties	78
6.8.4	Second-order Gear method's properties	82
6.8.5	Summary	83

6.9 Heun method.....	83
7. The discretized and linearized nodal voltage system in time step $v+1$ and iteration step $\mu+1$	85
7.1 Discretizing and linearizing the differential and nonlinear nodal system equations.....	85
7.2 Transient simulation flow	87
7.3 Discretization – circuit equivalent	88
7.4 Example for setting up a discretized and linearized nodal equation system	89
7.5 Nonlinearity and energy storage in the same element.....	91
7.5.1 Nonlinear capacitor.....	91
7.5.2 Nonlinear inductor	92
8. Index.....	95

1. Introduction

Circuit simulation and layout synthesis were the first areas of computer-aided design (CAD) of electronic circuits or electronic design automation (EDA), respectively. Circuit simulation, the subject of this compendium, comprises the numerical methods and CAD tools to analyze the behavior of an electronic circuit on the transistor level, which means its continuous voltage and current values in the operating point, over frequency, or over time. Circuit simulation aims to validate that a design meets its function requirements before producing costly silicon chips. An essential prerequisite of circuit simulation is the availability of a computer model of the circuit to be simulated. Electronic circuits on the circuit level are usually modeled as structural connections of lumped circuit elements like resistors R , capacitors C , inductors L , controlled sources, transistors T , etc., with connections ideal, i.e., without electrical properties. The resulting netlists represent a differential algebraic equation system (DAE):

$$\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), t) = \mathbf{0} \quad (1)$$

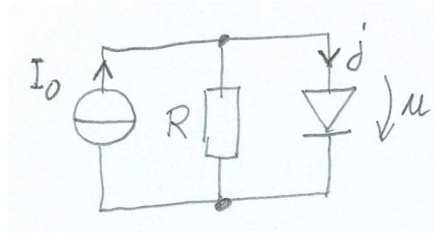
Complex elements like transistors represent complex behavior, again modeled by complete netlists of equivalent circuits for different simulation types. Templates for the circuit elements are given as library elements and expanded for a particular electronic circuit netlist. An electronic circuit and its corresponding DAE, represented by a netlist and library templates of circuit elements, have specific properties that allow efficient, standardized simulation methods to be set up. This includes three main tasks:

- How are the nonlinear circuit parts treated?
- How are the differential circuit parts treated?
- How are the solution equations set up?

1.1 Nonlinear circuit parts

Figure 1 shows a simple circuit with a nonlinear element, a diode.

Figure 1: Circuit with nonlinear diode.

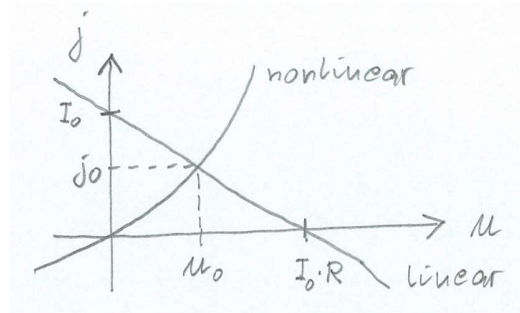


Applying Kirchhoff's current law at the top node of the circuit and introducing a simple nonlinear equation for the current-voltage relation at the diode yields the following equation (3) with a linear part on the left and a nonlinear part on the right. The solution of the nonlinear equation cannot be done analytically. In this simple case, one could find a graphical solution (u_0, j_0) according to Figure 2. In the general case, a numerical solution is created based on a systematic, iterative sequence of linearizations.

$$I_0 = j + \frac{u}{R} = I_S \cdot (e^{\frac{u}{V_T}} - 1) + \frac{u}{R} \quad (2)$$

$$\Leftrightarrow \underbrace{I_0 - \frac{u}{R}}_{\text{linear}} = \underbrace{I_S \cdot (e^{\frac{u}{V_T}} - 1)}_{\text{nonlinear}} \quad (3)$$

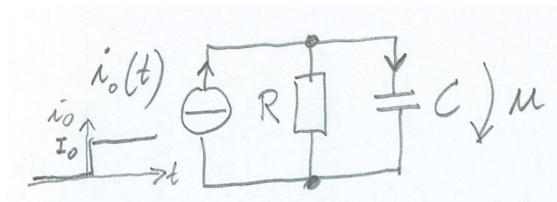
Figure 2: Solution of Eq. (3) is the intersection point of linear and nonlinear parts.



1.2 Differential circuit parts

Figure 3 shows a simple circuit with a capacitor representing a differential circuit part. The input is a step function.

Figure 3: Circuit with differential part from the capacitor.



Applying Kirchhoff's current law at the top node and inserting the differential equation of a capacitor yields the following DAE (4). It can be solved manually or symbolically, respectively, by applying a Laplace transformation of (4) into the complex frequency domain, then performing a partial fraction decomposition on the resulting polynomial quotient, followed by an inverse Laplace transformation on each partial fraction back to the time domain:

$$C \cdot \dot{u}(t) + \frac{1}{R} \cdot u(t) = i_0(t) \quad \circ \bullet \quad (4)$$

$$\dots \text{partial fraction decomposition} \dots \quad (5)$$

$$\bullet \circ \quad u(t) = I_0 \cdot R \cdot (1 - e^{-\frac{t}{R \cdot C}}) \quad (6)$$

This approach is too expensive for large systems and not possible when nonlinear circuit parts are present at the same time. In general, numerical integration methods are applied in this case.

1.3 In the following.

we will deal with solving the nonlinear equations from nonlinear circuit parts. First, one-dimensional bi-sectioning and the one-dimensional Newton approach will be discussed, followed by the multivariate Newton approach. An interpretation of the Newton-Raphson iteration with a circuit equivalent will be presented. Solving a nonlinear circuit represents a so-called DC simulation and is dedicated to finding the DC operating point of a circuit.

An AC simulation, or small-signal analysis, is dedicated to analyzing the frequency behavior around the DC operating point for small signals, i.e., amplitudes. This is done based on linearized circuit equivalent models around the DC operating point. The resulting mathematical problem is to solve a large, sparse linear equation system. Gaussian elimination and suboptimal pivoting will be discussed.

A TR simulation, or transient simulation, is dedicated to analyzing the transient behavior of a circuit, e.g., to the steady state after switching the input signals. Transient simulation requires numerical integration methods and linearization at the same time. We will discuss four simple numerical integration methods. These are the Forward Euler method, the Backward Euler method, the Trapezoidal method, and the second-order Gear method. Like Newton's approach to analyzing nonlinear circuits, we will interpret the numerical integration formulas by a circuit equivalent.

A computer-oriented circuit description most suitable for circuit simulation will be presented. This is the so-called modified nodal system. We will show how it can be set up directly from the netlist. It will apply to the circuit equivalent from the linearized circuit parts and the circuit equivalents from the differential circuit parts.

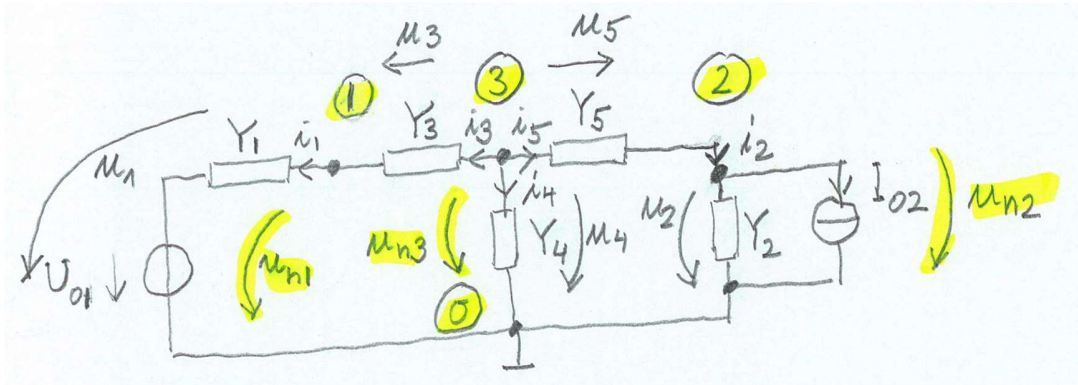
Setting up the modified nodal system requires the definition of a standard branch of the circuit. The index denoting a branch is κ (kappa). The iteration index is μ (mu), and the discrete time step is ν (nu). This explains the nickname "mu-nu-kappa lecture" of the course.

We will first treat setting up the modified nodal system. Then, we treat AC simulation and solve a linear equation system. This is followed by DC simulation and the Newton method. After that, we discuss TR simulation and numerical integration. The order in the design process differs: first, DC simulation, then AC and TR simulations.

2. Computer-based circuit representation and analysis

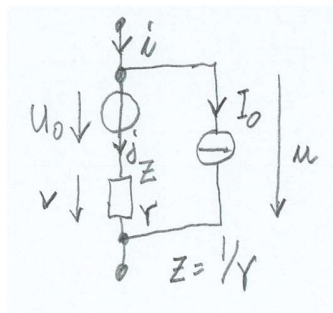
We will derive the nodal system of a circuit using the example circuit in Figure 4, built from a voltage source U_{01} , a current source I_{02} , and five admittances Y_1 to Y_5 .

Figure 4: Example circuit.



We define a **standard branch** (*Standardkante*) of a circuit as a serial connection of a voltage source U_0 and impedance Z , parallel with a current source I_0 , as shown in Figure 5.

Figure 5: Standard branch.



The following relations hold for the currents and voltages of the standard branch defined in Figure 5.

$$j = i - I_0 = Y \cdot (u - U_0) = Y \cdot v \quad (7)$$

$$v = u - U_0 = Z \cdot (i - I_0) = Z \cdot j \quad (8)$$

Please note that the impedance Z is a placeholder and can take the form (9) for a DC simulation, (10) for an AC simulation, and (11) for a transient simulation.

$$Z = R \quad (9)$$

$$Z = R + j\omega L \quad (10)$$

$$Z = L \cdot \frac{d}{dt}(\cdot) \quad (11)$$

With the definition of a standard branch, the example circuit in Figure 4 has five branches. The elements in the five branches have been indexed with a corresponding branch number. A circuit can be represented by a directed graph $G=(V, E)$ with the **vertex set** (*Knotenmenge*) V and the **edge set** (*Kantenmenge*) E .

$$V = \{\alpha | \alpha = 0, \dots, n\} \quad , \quad |V| = n + 1 \quad (12)$$

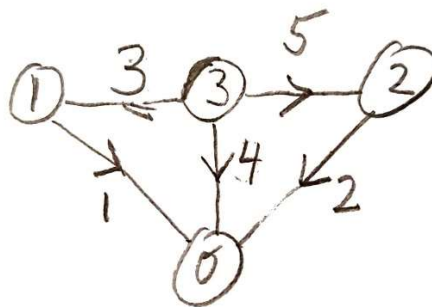
$$E = \{(\alpha, \beta) | \alpha, \beta \in V \wedge \alpha E \beta\} \quad , \quad |E| = k \quad (13)$$

There are $n+1$ vertices and k edges. An edge is a relation, i.e. connection, between two vertices, α and β . The graph of the example circuit has the following vertex and edge sets.

$$V = \{0, 1, 2, 3\} \quad , \quad E = \left\{ \underbrace{(1, 0)}_1, \underbrace{(2, 0)}_2, \underbrace{(3, 1)}_3, \underbrace{(3, 0)}_4, \underbrace{(3, 2)}_5 \right\} \quad (14)$$

The graph's graphical representation is shown in Figure 6.

Figure 6: Directed graph of example circuit in Figure 4.



We define the direction of an edge to have the same direction as the current in the corresponding circuit part.

The currents in the edges are collected in the so-called **branch current vector** (*Kantenstromvektor*) \mathbf{i} .

The voltages across the edges are collected in the so-called **branch voltage vector** (*Kantenspannungsvektor*) \mathbf{u} .

We also collect the differences in voltage potentials from all vertices to a pre-selected reference vertex 0 (usually the ground node) in the so-called **node voltage vector** (*Knotenspannungsvektor*) \mathbf{u}_n .

$$\mathbf{i} = [i_1 \ i_2 \ \dots \ i_k]^T \quad (15)$$

$$\mathbf{u} = [u_1 \ u_2 \ \dots \ u_k^T] \quad (16)$$

$$\mathbf{u}_n = [u_{n1} \ u_{n2} \ \dots \ u_{nn}] \quad (17)$$

The circuit elements in the standard branches are also collected in vectors and matrices.

These are the **branch current source vector** (*Kantenquellenstromvektor*) \mathbf{I}_0 ,
the **branch voltage source vector** (*Kantenquellenspannungsvektor*) \mathbf{U}_0 , and
the diagonal **branch admittance matrix** (*Kantenadmittanzmatrix*) \mathbf{Y} .

$$\mathbf{I}_0 = [I_{01} \ I_{02} \ \dots \ I_{0k}]^T \quad (18)$$

$$\mathbf{U}_0 = [U_{01} \ U_{02} \ \dots \ U_{0k}]^T \quad (19)$$

$$\mathbf{Y} = \text{diag}(Y_1, Y_2, \dots, Y_k) \quad (20)$$

2.1 Nodal voltage system

The system equations are established by setting up the following relations and inserting them into each other, i.e., KVL \rightarrow BCE \rightarrow KCL:

- Kirchhoff's current law (KCL) (*Kirchhoff'sches Stromgesetz, KI*) at all circuit nodes
- Branch constitutive equations (BCE), i.e., Ohm's law (*Ohmsches Gesetz, OG*) in all branches
- Kirchhoff's voltage law (KVL) (*Kirchhoff'sches Spannungsgesetz, KU*) for the branch voltages

2.1.1 Kirchhoff's current law (KCL) at all circuit nodes

The example circuit in Figure 4 has four nodes. Kirchhoff's current law (the sum of currents into a node is zero) is applied at each node. We define a current flowing out of the node as positive and a current flowing into the node as negative. This results in the following four equations.

$$\begin{array}{rcccl} \text{node } \downarrow / \text{edge } \rightarrow & 1: & 2: & 3: & 4: & 5: & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ \hline & & & & & & \end{array} \quad (21)$$

$$\begin{array}{r} 3: \\ 1: \\ 2: \\ 0: \end{array} \begin{array}{r} \\ i_1 \\ i_2 \\ -i_1 \\ -i_2 \end{array} \begin{array}{r} \\ \\ i_2 \\ -i_2 \end{array} \begin{array}{r} \\ \\ -i_3 \\ -i_3 \end{array} \begin{array}{r} \\ \\ \\ -i_4 \\ -i_4 \end{array} \begin{array}{r} \\ \\ \\ \\ -i_5 \end{array} \begin{array}{r} \\ \\ \\ \\ \\ = 0 \end{array}$$

One of these equations is linearly dependent on the remaining equations. E.g., adding the first three equations equals the last equation multiplied by -1. One equation can, therefore, be skipped. The equation for the ground node is ignored because it has the most connections. Skipping this equation keeps the density of the remaining equation system sparse. Collecting the coefficients of the other equations in the so-called **node matrix** (*Knotenmatrix*) \mathbf{A} , i.e.,

$$\mathbf{A} = \begin{array}{r} 3: \\ 1: \\ 2: \end{array} \begin{bmatrix} 1: & 2: & 3: & 4: & 5: \\ 0 & 0 & +1 & +1 & +1 \\ +1 & 0 & -1 & 0 & 0 \\ 0 & +1 & 0 & 0 & -1 \end{bmatrix} \quad (22)$$

we can formulate **Kirchhoff's current law** in matrix-vector notation for any circuit:

$$\text{KCL: } \mathbf{A} \cdot \mathbf{i} = \mathbf{0} \quad (23)$$

\mathbf{A} is also called the **incidence matrix** (*Inzidenzmatrix*), which describes the node-edge incidences.

2.1.2 Branch constitutive equations (BCE), i.e. Ohm's law in all branches

Next, we will formulate Ohm's law in all circuit branches. For the example circuit, this results in:

$$\begin{aligned} i_1 &= Y_1 \cdot (u_1 - U_{01}) \\ i_2 - I_{02} &= Y_2 \cdot u_2 \\ i_3 &= Y_3 \cdot u_3 \\ i_4 &= Y_4 \cdot u_4 \\ i_5 &= Y_5 \cdot u_5 \end{aligned} \quad (24)$$

Using the definitions for currents, voltages, sources, and admittances in branches in (15), (16), and (18) to (20), we can formulate the branch constitutive equations in matrix-vector notation for any circuit:

$$\text{BCE: } \mathbf{i} - \mathbf{I}_0 = \mathbf{Y} \cdot (\mathbf{u} - \mathbf{U}_0) \quad (25)$$

2.1.3 Kirchhoff's voltage law (KVL) for all branch voltages

In the third step, the branch voltages will be formulated based on the node voltages, applying Kirchhoff's voltage law. For the example circuit, we obtain

$$\begin{aligned} u_1 &= u_{n1} & 1: & \begin{bmatrix} 0 & +1 & 0 \\ 0 & 0 & +1 \\ +1 & -1 & 0 \\ +1 & 0 & 0 \\ +1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} u_{n3} \\ u_{n1} \\ u_{n2} \end{bmatrix} \\ u_2 &= u_{n2} & 2: & \\ u_3 &= u_{n3} - u_{n1} & \Leftrightarrow \mathbf{u} = & 3: \\ u_4 &= u_{n3} & 4: & \\ u_5 &= u_{n3} - u_{n2} & 5: & \end{aligned} \quad (26)$$

On the right, the relation between branch voltages and node voltages is written in matrix-vector notation. We recognize that the resulting matrix is the transposed incidence matrix. We state without proof that this relation can be written in matrix-vector notation for any circuit:

$$\text{KVL: } \mathbf{u} = \mathbf{A}^T \cdot \mathbf{u}_n \quad (27)$$

We insert (27) into (25) to replace \mathbf{u} with \mathbf{u}_n . Then, we insert (25) into (23) to replace \mathbf{i} with \mathbf{u}_n .

$$\underbrace{\begin{array}{ccc} \text{KVL} & \rightarrow & \text{BCE} & \rightarrow & \text{KCL} \\ \mathbf{u}_n \rightarrow \mathbf{u} & & \mathbf{u} \rightarrow \mathbf{i} & & \mathbf{F}(\mathbf{i}) = \mathbf{0} \end{array}}_{\mathbf{F}(\mathbf{u}_n) = \mathbf{0}} \quad (28)$$

This results in the following:

$$\mathbf{A} \cdot [\mathbf{Y} \cdot (\mathbf{A}^T \cdot \mathbf{u}_n - \mathbf{U}_0) + \mathbf{I}_0] = \mathbf{0} \quad (29)$$

The given circuit determines everything but the node voltage vector. Reformulation of (29) yields the following equation system with the node voltages as unknowns/system coordinates:

$$\underbrace{\mathbf{A} \cdot \mathbf{Y} \cdot \mathbf{A}^T}_{\mathbf{Y}_n} \cdot \mathbf{u}_n = \underbrace{\mathbf{A} \cdot (\mathbf{Y} \cdot \mathbf{U}_0 - \mathbf{I}_0)}_{\mathbf{I}_n} \quad (30)$$

\mathbf{Y}_n is called the **nodal admittance matrix** (*Knotenadmittanzmatrix*).

\mathbf{I}_n is called the **nodal current source vector** (*Knotenquellenstromvektor*).

(30) represents a circuit's nodal voltage system (*Knotenspannungssystem*). The nodal voltage system is a unique description of the circuit. Its solution provides the node voltages from which any other current or voltage in the circuit can be derived.

A significant advantage of the nodal voltage system is that it does not have to be set up by multiplying matrices and vectors as given in (29) but can be set up directly from the circuit netlist. We will illustrate this by performing the three insertions (27) \rightarrow (25) \rightarrow (23) for the circuit example in [Figure 4](#), i.e., by inserting (26) \rightarrow (24) \rightarrow (21). This results in:

$$\begin{aligned} Y_3 \cdot (u_{n3} - u_{n1}) + Y_4 \cdot u_{n3} + Y_5 \cdot (u_{n3} - u_{n2}) &= 0 \\ Y_1 \cdot (u_{n1} - U_{01}) - Y_3 \cdot (u_{n3} - u_{n1}) &= 0 \\ Y_2 \cdot u_{n2} + I_{02} - Y_5 \cdot (u_{n3} - u_{n2}) &= 0 \end{aligned} \quad (31)$$

and in matrix-vector notation:

$$\begin{array}{l}
 3: \\
 1: \\
 2:
 \end{array}
 \begin{bmatrix}
 Y_3 + Y_4 + Y_5 & -Y_3 & -Y_5 \\
 -Y_3 & Y_1 + Y_3 & 0 \\
 -Y_5 & 0 & Y_2 + Y_5
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 u_{n3} \\
 u_{n1} \\
 u_{n2}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 Y_1 \cdot U_{01} \\
 -I_{02}
 \end{bmatrix}
 \quad (32)$$

Please note that (32) has a permutation in the ordering of the node voltage vector.

Comparing (32) with [Figure 4](#) shows the following patterns:

A diagonal element (i,i) of the nodal admittance matrix shows the sum of all admittances in branches connected to the respective node i.

A non-diagonal element (i,j) of the nodal admittance shows the negative admittance in the branch between the respective nodes i and j.

A right-hand-side element i shows the branch current source; the sign is negative if the current points away from the node and positive if it points toward the node. The current in branch 1 is obtained from transforming the series connection of resistor and voltage source into a parallel connection of admittance and current source with the value given in (32) and pointing toward node 1.

Each circuit element “stamps” specific patterns into the nodal voltage system. Therefore, the nodal voltage system can be set up uniquely and immediately from a given circuit netlist. Later on, we will interpret the iterative and successive equations resulting from the Newton method for linearization and from numerical integration methods by equivalent circuits. These equivalent circuits can then be “stamped” into the nodal voltage system in the same way as we have discovered now.

We need another step leading to the modified nodal system for now. This step is necessary if a standard branch has a short instead of an admittance. The admittance would be infinite, i.e., $Y \rightarrow \infty$, and an infinite number is not allowed in a linear equation system with real numbers. This is treated in the following.

2.2 Modified nodal voltage system/modified nodal analysis (MNA)

In the case of a standard branch according to [Figure 5](#) with $Y \rightarrow \infty$, the nodal voltage system cannot be set up as described above. Look at the KCL (23), the BCE (25), and the KVL (27): Where can we change something to avoid having a symbolic number “ ∞ ”, which is not allowed in a numerical representation?

Exactly, it is the BCE. We can write Ohm’s law with voltage as impedance times current, where the impedance can become zero in a numerical environment.

To this end, we partition the branches into two types: the “regular” **Y-branches** (*Y-Kanten*), where Ohm’s law is set up as in (25), and the **Z-branches** (*Z-Kanten*), where Ohm’s law is set up with voltage as impedance times current. Z-branches must be introduced in the case of $Y \rightarrow \infty$; they may be used anytime. The partitioning of branches entails partitioning the branch current (source) vector, the branch voltage (source) vector, and the

incidence matrix into Y-parts and Z-parts. In addition, the branch admittance matrix is set up only for Y-branches. For the Z-branches, a **branch impedance matrix** is set up instead.

$$\mathbf{i} = \begin{bmatrix} \mathbf{i}_Y \\ \mathbf{i}_Z \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_Y \\ \mathbf{u}_Z \end{bmatrix} \quad \mathbf{I}_0 = \begin{bmatrix} \mathbf{I}_{0Y} \\ \mathbf{I}_{0Z} \end{bmatrix} \quad \mathbf{U}_0 = \begin{bmatrix} \mathbf{U}_{0Y} \\ \mathbf{U}_{0Z} \end{bmatrix} \quad (33)$$

$$\mathbf{A} = [\mathbf{A}_Y \ \mathbf{A}_Z], \quad \mathbf{Y}_Y, \ \mathbf{Z}_Z$$

After the partitioning, KCL, BCE, and KVL take the following form.

$$\mathbf{KCL}_{YZ} : \quad [\mathbf{A}_Y \ \mathbf{A}_Z] \cdot \begin{bmatrix} \mathbf{i}_Y \\ \mathbf{i}_Z \end{bmatrix} = \mathbf{0} \quad (34)$$

$$\mathbf{BCE}_Y : \quad \mathbf{i}_Y - \mathbf{I}_{0Y} = \mathbf{Y}_Y \cdot (\mathbf{u}_Y - \mathbf{U}_{0Y}) \quad (35)$$

$$\mathbf{BCE}_Z : \quad \mathbf{u}_Z - \mathbf{U}_{0Z} = \mathbf{Z}_Z \cdot (\mathbf{i}_Z - \mathbf{I}_{0Z}) \quad (36)$$

$$\mathbf{KVL}_Y : \quad \mathbf{u}_Y = \mathbf{A}_Y^T \cdot \mathbf{u}_n \quad (37)$$

$$\mathbf{KVL}_Z : \quad \mathbf{u}_Z = \mathbf{A}_Z^T \cdot \mathbf{u}_n \quad (38)$$

We insert (37) into (35) to replace \mathbf{u}_Y with \mathbf{u}_n . Then we insert (35) into (34) to replace \mathbf{i}_Y with \mathbf{u}_n :

$$\underbrace{\begin{array}{ccc} \mathbf{KVL}_Y & \rightarrow & \mathbf{BCE}_Y & \rightarrow & \mathbf{KCL}_{YZ} \\ \mathbf{u}_n \rightarrow \mathbf{u}_Y & & \mathbf{u}_Y \rightarrow \mathbf{i}_Y & & \mathbf{F}_1(\mathbf{i}_Y, \mathbf{i}_Z) = \mathbf{0} \end{array}}_{\mathbf{F}_1(\mathbf{u}_n, \mathbf{i}_Z) = \mathbf{0}} \quad (39)$$

This yields:

$$\mathbf{A}_Y \cdot [\mathbf{Y}_Y \cdot (\mathbf{A}_Y^T \cdot \mathbf{u}_n - \mathbf{U}_{0Y}) + \mathbf{I}_{0Y}] + \mathbf{A}_Z \cdot \mathbf{i}_Z = \mathbf{0} \quad (40)$$

In addition, we insert (38) into (36) to replace \mathbf{u}_Z with \mathbf{u}_n :

$$\underbrace{\begin{array}{ccc} \mathbf{KVL}_Z & \rightarrow & \mathbf{BCE}_Z \\ \mathbf{u}_n \rightarrow \mathbf{u}_Z & & \mathbf{F}_2(\mathbf{u}_Z, \mathbf{i}_Z) = \mathbf{0} \end{array}}_{\mathbf{F}_2(\mathbf{u}_n, \mathbf{i}_Z) = \mathbf{0}} \quad (41)$$

This results in:

$$\mathbf{A}_Z^T \cdot \mathbf{u}_n - \mathbf{U}_{0Z} = \mathbf{Z}_Z \cdot (\mathbf{i}_Z - \mathbf{I}_{0Z}) \quad (42)$$

(40) and (42) establish a linear equation system with the node voltages \mathbf{u}_n and the branch currents \mathbf{i}_Z in the Z-branches as unknowns:

$$\begin{bmatrix} \mathbf{A}_Y \cdot \mathbf{Y}_Y \cdot \mathbf{A}_Y^T & \mathbf{A}_Z \\ \mathbf{A}_Z^T & -\mathbf{Z}_Z \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_n \\ \mathbf{i}_Z \end{bmatrix} = \begin{bmatrix} \mathbf{A}_Y \cdot (\mathbf{Y}_Y \cdot \mathbf{U}_{0Y} - \mathbf{I}_{0Y}) \\ \mathbf{U}_{0Z} - \mathbf{Z}_Z \cdot \mathbf{I}_{0Z} \end{bmatrix} \quad (43)$$

$$\begin{bmatrix} \mathbf{Y}_{nY} & \mathbf{A}_Z \\ \mathbf{A}_Z^T & -\mathbf{Z}_Z \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_n \\ \mathbf{i}_Z \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{nY} \\ \mathbf{U}_{ZZ} \end{bmatrix}$$

\mathbf{Y}_{nY} denotes the **nodal admittance matrix for Y-branches only**.

\mathbf{I}_{nY} represents the **nodal current source vector for Y-branches only**.

\mathbf{U}_{ZZ} denotes the **branch aggregate voltage source vector in Z-branches only**.

This is the so-called modified nodal system and forms the basis for the so-called **modified nodal analysis (MNA)** (*modifizierte Knotenspannungsanalyse*). Please note that every Z-branch adds one more unknown to the nodal systems: the corresponding Z-branch current.

In the following, we will set up the modified nodal system for the circuit example in [Figure 4](#), assuming that the admittance in branch 1 becomes infinite, i.e., $Y_1 \rightarrow \infty$, i.e., only a voltage source U_{01} in branch 1. We obtain the following sets of Y-branches and Z-branches.

Y-branches: 2, 3, 4, 5

Z-branches: 1

Accordingly, (23) with (22) is partitioned into a part for the Y-branches and a part for the Z-branch:

$$\begin{bmatrix} 0 & +1 & +1 & +1 \\ 0 & -1 & 0 & 0 \\ +1 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} i_2 \\ i_3 \\ i_4 \\ i_5 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot i_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (44)$$

$$\mathbf{A}_Y \cdot \mathbf{i}_Y + \mathbf{A}_Z \cdot \mathbf{i}_Z = \mathbf{0}$$

(44) is equal to (23) and (22), with a distinction between the Y-branches and Z-branches.

The branch constitutive equations are set up for the Y-branches only in the form of (35).

$$\begin{bmatrix} i_2 \\ i_3 \\ i_4 \\ i_5 \end{bmatrix} - \begin{bmatrix} I_{02} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} Y_2 & 0 & 0 & 0 \\ 0 & Y_3 & 0 & 0 \\ 0 & 0 & Y_4 & 0 \\ 0 & 0 & 0 & Y_5 \end{bmatrix} \cdot \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} \quad (45)$$

$$\mathbf{i}_Y - \mathbf{I}_{0Y} = \mathbf{Y}_Y \cdot (\mathbf{u}_Y - \mathbf{U}_{0Y})$$

Compared to (24), the Z-branch 1 is skipped in (42).

For the Z-branches, the branch constitutive equations are set up in the form of (36):

$$\begin{aligned} u_1 - U_{01} &= \frac{1}{Y_1} \cdot i_1 \\ \mathbf{u}_Z - \mathbf{U}_{0Z} &= \mathbf{Z}_Z \cdot (\mathbf{i}_Z - \mathbf{I}_{0Z}) \end{aligned} \quad (46)$$

Then, the Kirchhoff voltage law is applied to replace the branch voltages with the node voltages. The Y-branches are obtained according to (37):

$$\begin{aligned} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & +1 \\ +1 & -1 & 0 \\ +1 & 0 & 0 \\ +1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} u_{n3} \\ u_{n1} \\ u_{n2} \end{bmatrix} \\ \mathbf{u}_Y &= \mathbf{A}_Y^T \cdot \mathbf{u}_n \end{aligned} \quad (47)$$

This is the same as in (26); only the Z-branch 1 has been left out. This Z-branch is formulated separately according to (38).

$$\begin{aligned} u_1 &= [0 \ 1 \ 0] \cdot \begin{bmatrix} u_{n3} \\ u_{n1} \\ u_{n2} \end{bmatrix} \\ \mathbf{u}_Z &= \mathbf{A}_Z^T \cdot \mathbf{u}_n \end{aligned} \quad (48)$$

The combination of (47) and (48) corresponds to (26). Insertion according to (39) and (41) then results in:

$$\begin{aligned} Y_3 \cdot (u_{n3} - u_{n1}) + Y_4 \cdot u_{n3} + Y_5 \cdot (u_{n3} - u_{n2}) &= 0 \\ -Y_3 \cdot (u_{n3} - u_{n1}) + i_1 &= 0 \\ Y_2 \cdot u_{n2} + I_{02} - Y_5 \cdot (u_{n3} - u_{n2}) &= 0 \\ u_{n1} - \frac{1}{Y_1} \cdot i_1 - U_{01} &= 0 \end{aligned} \quad (49)$$

The first three equations correspond to (39), and the last corresponds to (41). Please note that we have written (49) symbolically. At this point, we ignore that $1/Y_1$ will be zero according to the initial assumption. But even if it was not zero, we could set up this modified nodal system. (49) can be written in matrix-vector notation:

$$\begin{bmatrix} Y_3 + Y_4 + Y_5 & -Y_3 & -Y_5 & 0 \\ -Y_3 & Y_3 & 0 & 1 \\ -Y_5 & 0 & Y_2 + Y_5 & 0 \\ 0 & 1 & 0 & -\frac{1}{Y_1} \end{bmatrix} \cdot \begin{bmatrix} u_{n3} \\ u_{n1} \\ u_{n2} \\ i_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -I_{02} \\ U_{01} \end{bmatrix} \quad (50)$$

This example illustrates that the modified nodal system can be set up directly from the given netlist without intermediate operations. The rules are listed in Sec. 2.4. Please note that the given rules are element-wise for the equation system. They can also be given element-wise for the circuit. Then, each type of circuit element “stamps” into the equation system in a specific way.

2.3 Integration of controlled sources into the nodal voltage system

Controlled sources are standard circuit elements. Generic equations describing the control of current sources and voltage sources by branch voltages or branch currents are:

$$\begin{aligned} \mathbf{U}_0 &= \mathbf{U}_* + \mathbf{N} \cdot \mathbf{u} + \mathbf{R} \cdot \mathbf{i} \\ \mathbf{I}_0 &= \mathbf{I}_* + \mathbf{G} \cdot \mathbf{u} + \mathbf{M} \cdot \mathbf{i} \end{aligned} \quad (51)$$

These equations for controlled sources replace the constant sources in the standard branches defined in Figure 5. The constant voltage and current parts are indexed with an asterisk. Please note that we assume there are no current sources and no voltage sources in controlling branches!

The branch voltages and branch currents in (51) need to be transformed such that controlled sources depend on node voltages:

$$\begin{aligned} \mathbf{U}_0 &= \mathbf{U}_* + \mathbf{N}_n \cdot \mathbf{u}_n \\ \mathbf{I}_0 &= \mathbf{I}_* + \mathbf{G}_n \cdot \mathbf{u}_n \end{aligned} \quad (52)$$

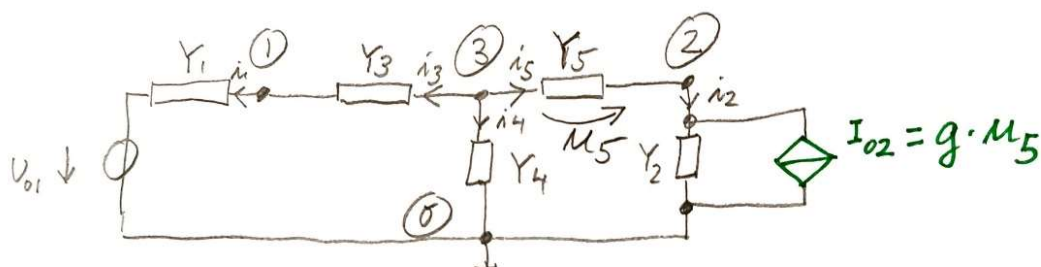
Inserting (52) into the nodal system (30) results in:

$$[\mathbf{A} \cdot \mathbf{Y} \cdot \mathbf{A}^T - \mathbf{A} \cdot \mathbf{Y} \cdot \mathbf{N}_n + \mathbf{A} \cdot \mathbf{G}_n] \cdot \mathbf{u}_n = \mathbf{A} \cdot (\mathbf{Y} \cdot \mathbf{U}_* - \mathbf{I}_*) \quad (53)$$

Controlled sources in a modified nodal system are integrated analogously.

For example, let us assume that the current source I_{02} of the example circuit in Figure 4 is controlled by the branch voltage in branch 5, as illustrated in Figure 7.

Figure 7: Example circuit with controlled source.



First, we have to express u_5 by node voltages. This gives:

$$I_{02} = g \cdot (u_{n3} - u_{n2}) \quad (54)$$

Inserting (54) into the right side of (50) and bringing the coefficients of u_{n2} and u_{n3} on the left side results in:

$$\begin{bmatrix} Y_3 + Y_4 + Y_5 & -Y_3 & -Y_5 & 0 \\ -Y_3 & Y_3 & 0 & 1 \\ +g - Y_5 & 0 & Y_2 + Y_5 - g & 0 \\ 0 & 1 & 0 & -\frac{1}{Y_1} \end{bmatrix} \cdot \begin{bmatrix} u_{n3} \\ u_{n1} \\ u_{n2} \\ i_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -I_{02} \\ U_{01} \end{bmatrix} \quad (55)$$

2.4 Setup rules for the modified nodal system

Overall, the following setup rules for the modified nodal system with the node admittance matrix for the Y-branches, the incidence matrix for the Z-branches, the impedance matrix of the Z-branches, as well as the node current source vector, and the branch voltage source vector can be formulated as follows.

$$\begin{bmatrix} \mathbf{Y}_{nY} & \mathbf{A}_Z \\ \mathbf{A}_Z^T & -\mathbf{Z}_Z \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_n \\ \mathbf{i}_Z \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{nY} \\ \mathbf{U}_{ZZ} \end{bmatrix}$$

\mathbf{Y}_{nY} : $Y_{nY,ii}$ (diagonal element): sum of admittances from y-branches at circuit node i
 $Y_{nY,ij} = Y_{nY,ji}$, $i \neq j$: negative admittance from y-branches between circuit nodes i and j

\mathbf{A}_Z : (branch direction = direction of Z-branch current)
 $A_{z,ij} = +1$ if Z-branch j leaves node i
 $A_{z,ij} = -1$ if Z-branch j arrives at node i

$\mathbf{Z}_z = \text{diag}(\mathbf{Z}_{zi})$: Z_{zi} : impedance in Z-branch i

\mathbf{I}_{nY} : $I_{nY,i}$: sum of current sources from Y-branches at node i (voltage sources with serial impedance are transformed into current sources with parallel admittance)
 current source from Y-branch arriving at node i: +
 current source from Y-branch leaving node i: -

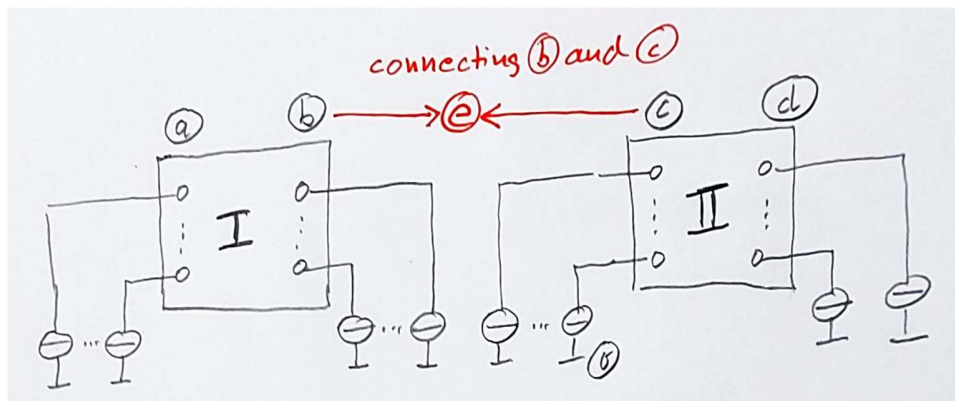
\mathbf{U}_{ZZ} : $U_{ZZ,i}$: sum of voltage sources in Z-branch i (current sources with parallel admittance are transformed in voltage sources with serial impedance)
 direction of voltage source same as direction of Z-branch i: +
 direction of voltage source opposite to direction of Z-branch i: -

Controlled sources: include the equation for the source on the right side; insert the controlling equation; move corresponding terms of unknowns of the equation system to the matrix on the left;

2.5 Connecting subsystems

Assume two subsystems I and II, as illustrated in Figure 8.

Figure 8: Two subsystems, I, with node sets a and b, and II, with node sets c and d, to be connected at the node sets b and c (with an equal number of nodes).



The nodal systems for both subsystems are written in terms of the node sets a to d. They are written into one nodal system for the nodal voltages of the four node sets. There are entries in the nodal admittance matrices and the nodal current source vectors that are not further detailed as the inside of the systems is unknown. But as the two systems are separate, there is no connection between them; hence, there are no entries in the overall nodal admittance matrix between a and b of system I and c and d of system II:

$$\begin{bmatrix} Y_{naa} & Y_{nab} & 0 & 0 \\ Y_{nba} & Y_{nbb} & 0 & 0 \\ 0 & 0 & Y_{ncc} & Y_{ncd} \\ 0 & 0 & Y_{ndc} & Y_{ndd} \end{bmatrix} \cdot \begin{bmatrix} u_{na} \\ u_{nb} \\ u_{nc} \\ u_{nd} \end{bmatrix} = \begin{bmatrix} I_{na} \\ I_{nb} \\ I_{nc} \\ I_{nd} \end{bmatrix} \quad (56)$$

What happens with the nodal system when connecting the two subsystems? The rules for setting up the nodal system say that the diagonal of the nodal admittance matrix sums all admittances at a node. When two nodes of sets b and c are connected, the resulting node e has the sum of all admittances connected to node b in system I and node c in system II.

An off-diagonal element sums all admittances between two nodes. Therefore, a node in set e sums all admittances to nodes to which the respective node in b is connected in system I and the nodes to which the respective node in c is connected in system II. Overall, the two admittance matrices for node sets b and c can simply be added.

Analogously, the nodal current source vector summarizes all current sources connected to a node. Hence, all current sources connected to a node in set b and all current sources connected to the corresponding node in set

c are now connected to the merged node in set e. This corresponds to adding the nodal current source vectors of nodes b and c.

Overall, we get for a subsystem connection:

$$\begin{bmatrix} \mathbf{Y}_{naa} & \mathbf{Y}_{nab} & \mathbf{0} \\ \mathbf{Y}_{nba} & \mathbf{Y}_{nbb} + \mathbf{Y}_{ncc} & \mathbf{Y}_{ncd} \\ \mathbf{0} & \mathbf{Y}_{ndc} & \mathbf{Y}_{ndd} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_{na} \\ \mathbf{u}_{ne} = \mathbf{u}_{nb} = \mathbf{u}_{nc} \\ \mathbf{u}_{nd} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{na} \\ \mathbf{I}_{nb} + \mathbf{I}_{nc} \\ \mathbf{I}_{nd} \end{bmatrix} \quad (57)$$

2.6 Advantages of the nodal voltage system

It turns out that the nodal voltage system is “the” computer-oriented system description.

- The setup rules are simple; the equation system can be set up directly from the circuit netlist.
- No tree definition, as in other types of system coordinates, is necessary.
- The reference node is simply the ground node of the circuit.
- The resulting system matrix is sparse. (Without controlled sources, it would be symmetric.)
- The system matrix is diagonal-dominant. (This is due to the setup rules: every element in a row or column is also part of the sum in the diagonal.)
- The setup complies with system theory: controlled sources and n-terminal devices are included in the nodal admittance matrix and the nodal current source vector; n-port networks are included in the branch constitutive equations.
- The nodal voltage system is technically interpretable. Section 2.5 has illustrated the connection of systems. A reduction of a system will be described in Secs. 3.5 and 3.6.

2.7 Modified nodal system by triangular decomposition of sparse network tableau

There is another way to get to the modified nodal system. Let us first write down (34)-(38) in a different way:

$$\begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Y^T \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Z^T \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_Y & \mathbf{A}_Z & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{Z}_Z & \mathbf{0} \\ \mathbf{Y}_Y & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_Y \\ \mathbf{u}_Z \\ \mathbf{i}_Y \\ \mathbf{i}_Z \\ \mathbf{u}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{Z}_Z \cdot \mathbf{I}_{0Z} - \mathbf{U}_{0Z} \\ \mathbf{Y}_Y \cdot \mathbf{U}_{0Y} - \mathbf{I}_{0Y} \end{bmatrix} \quad (58)$$

\mathbf{I} stands for the identity matrix. The first two rows represent the KVL for Y-branches and Z-branches from (37) and (38). The third row represents the KCL partitioned according to currents in Y-branches and Z-branches from (34). The last two rows represent the BCE for Z-branches and Y-branches from (35) and (36).

(58) concatenates all unknown system coordinates, i.e., the branch voltages and the branch currents, partitioned into Y-branches and Z-branches, and the node voltages in a particular order.

Instead of the symbolic insertions along node voltages into branch voltages into branch currents, we now perform a triangular decomposition of the system (58) into an upper block-triangular form similar to Gaussian elimination.

We eliminate the $-\mathbf{I}$ entry in row 4 by subtracting row 2 from row 4. We eliminate the \mathbf{Y}_Y entry in row 5 by adding \mathbf{Y}_Y times row 1 to row 5. This results in:

$$\begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Y^T \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Z^T \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_Y & \mathbf{A}_Z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_Z & -\mathbf{A}_Z^T \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{Y}_Y \cdot \mathbf{A}_Y^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_Y \\ \mathbf{u}_Z \\ \mathbf{i}_Y \\ \mathbf{i}_Z \\ \mathbf{u}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{Z}_Z \cdot \mathbf{I}_{0Z} - \mathbf{U}_{0Z} \\ \mathbf{Y}_Y \cdot \mathbf{U}_{0Y} - \mathbf{I}_{0Y} \end{bmatrix} \quad (59)$$

In (59), we eliminate the $-\mathbf{I}$ entry in row 5 by multiplying row 5 with \mathbf{A}_Y and adding row 3 to it. This finally produces the upper triangular form:

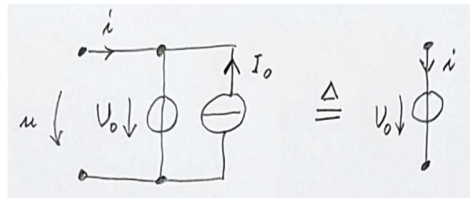
$$\begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Y^T \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Z^T \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_Y & \mathbf{A}_Z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_Z & -\mathbf{A}_Z^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_Z & \mathbf{A}_Y \cdot \mathbf{Y}_Y \cdot \mathbf{A}_Y^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}_Y \\ \mathbf{u}_Z \\ \mathbf{i}_Y \\ \mathbf{i}_Z \\ \mathbf{u}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{Z}_Z \cdot \mathbf{I}_{0Z} - \mathbf{U}_{0Z} \\ \mathbf{A}_Y \cdot (\mathbf{Y}_Y \cdot \mathbf{U}_{0Y} - \mathbf{I}_{0Y}) \end{bmatrix} \quad (60)$$

The last two rows of (60) represent the modified nodal system from (43) with a switched ordering of system coordinates and a switch in the sign of the last but one row in (60).

2.8 Source transformations

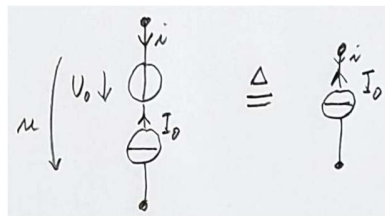
Source transformations will be given in the following, which may be helpful in the setup of the modified nodal system.

Figure 9: Parallel voltage and current source \rightarrow voltage source.



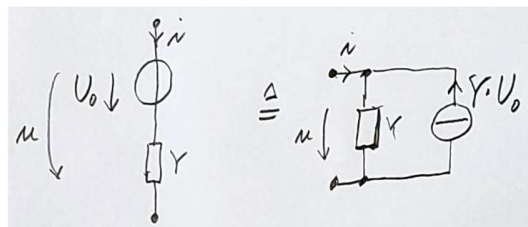
The voltage u in **Figure 9** is determined by the voltage source U_0 , i.e., $u=U_0$. The current i depends on the rest of the circuit, i.e., $i=f(\text{circuit})$. Hence, the current source I_0 has no influence and can be deleted.

Figure 10: Serial voltage and current source \rightarrow current source.



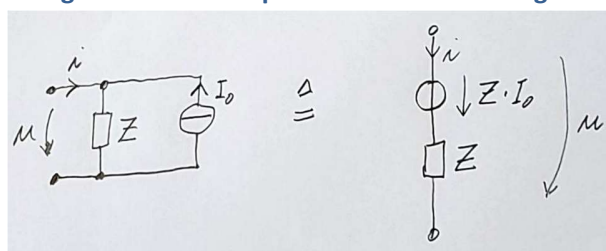
The current i in **Figure 10** is determined by the current source I_0 , i.e., $i=-I_0$. The voltage u depends on the rest of the circuit, i.e., $u=f(\text{circuit})$. Hence, the voltage source U_0 has no influence and can be deleted.

Figure 11: Serial voltage source and admittance \rightarrow parallel current source and admittance.



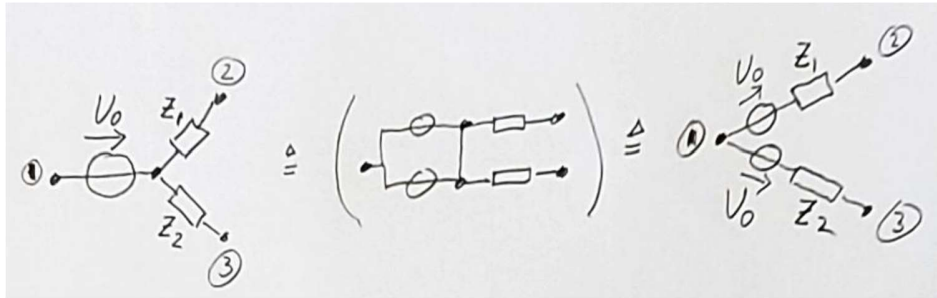
The current i for the two circuits in **Figure 11** is identical, i.e., $i = Y \cdot (u - U_0)$, as the circuit on the right provides.

Figure 12: Parallel voltage source and impedance \rightarrow serial voltage source and impedance.



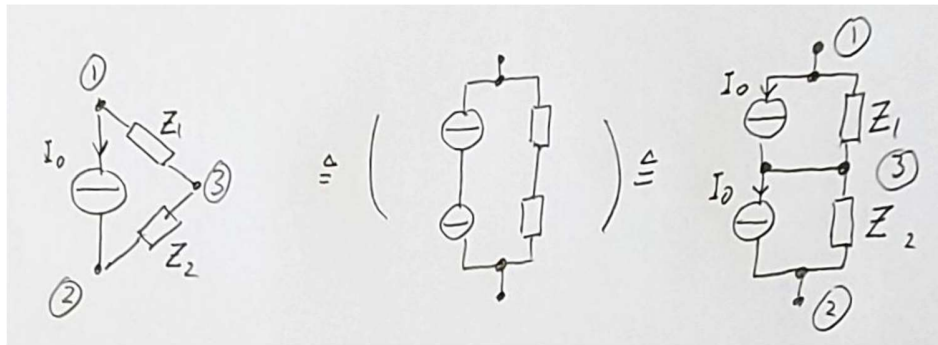
The voltage u for the two circuits in **Figure 12** is identical, i.e., $u = Z \cdot (i + I_0)$, as the circuit on the right provides.

Figure 13: Avoiding extra nodes/branches in a star configuration with a voltage source.



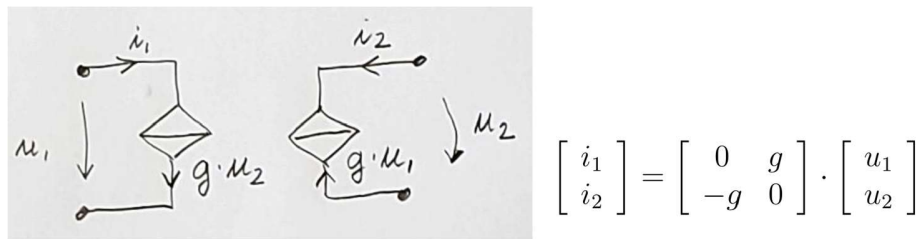
The voltage source at the left of **Figure 13** can be copied in parallel (circuit in the middle); the voltage from node 1 to the center node stays the same. As the voltage at the center node stays constant by the two identical voltage sources, the center node can be split, which yields two standard branches, as shown on the right.

Figure 14: Avoiding extra nodes/branches in a mesh configuration with a current source.



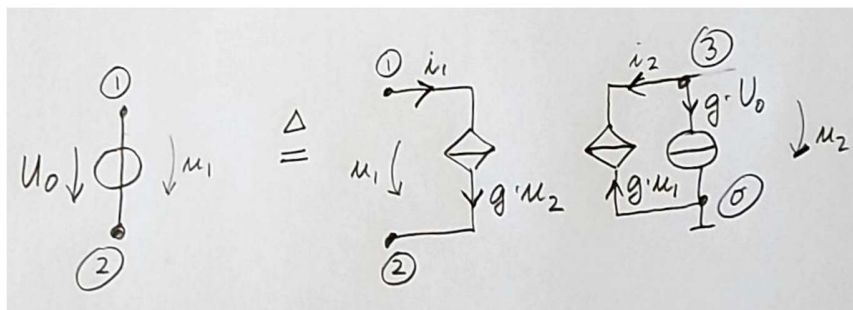
The current source on the left of **Figure 14** can be copied in a series (circuit in the middle); the current between nodes 1 and 2 stays the same. As the current at the node between the two identical current sources is constant, this node can be connected to the node between the two impedances without any current flowing from the left branch to the right branch. This allows the circuit on the right.

Figure 15: Gyrator.



The equation system and circuit equivalent in **Figure 15** define a gyrator.

Figure 16: Avoiding a Z-branch for a standard branch with only a voltage source by a gyrator and additional node.

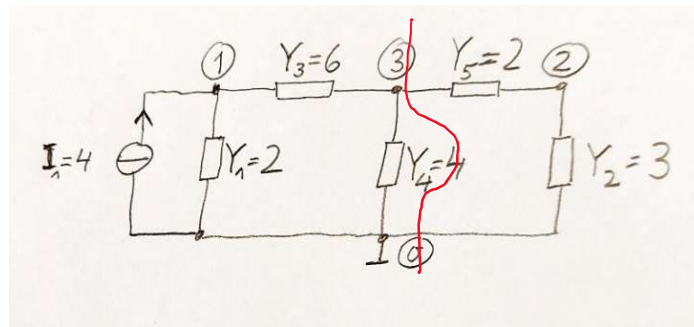


The gyrator from **Figure 15** has been inserted in **Figure 16**. At the additional node 3, a current source with value $g \cdot U_0$ has been inserted. At node 3, we obtain $i_2 = -g \cdot u_1 = -g \cdot U_0$. Hence, $u_1 = U_0$, which corresponds to the original edge. In both circuits on the left and right side **Figure 16**, we have an additional unknown in the nodal system. In the original case, it is a Z-branch with the current through U_0 . In the case of the gyrator equivalent circuit, it is a Y-branch with an additional node.

3. AC analysis

An AC analysis (*Wechselstrom-Analyse*) starts from the DC operating point (see Sec. 4) of a linear circuit or a linearized circuit modeled by the corresponding small-signal equivalent circuit (*Kleinsignal-Ersatzschaltbild*). The input to the circuit in AC analysis is a sine/cosine defined by its amplitude and frequency. The output is another sine/cosine with different amplitude, frequency, and phase. System equations of an AC analysis are linear equation systems with complex numbers. AC analysis, therefore, leads to the mathematical problem of solving a linear equation system. A typical method to solve a linear equation system is the Gaussian elimination, i.e., the LU decomposition (*LR-Zerlegung*). In the tutorial, we will treat an example of setting up the linear equation system for an AC analysis. In the following, we will review the Gaussian elimination. This will be done using the following artificial circuit example in **Figure 17**.

Figure 17: Circuit example for AC analysis.



Please note that we avoid complex numbers in this example to keep things simple. The input current's real number refers to the input sine's amplitude for a specific frequency not considered here. As there are no complex numbers, the output will have the same frequency, no phase shift, and a different amplitude.

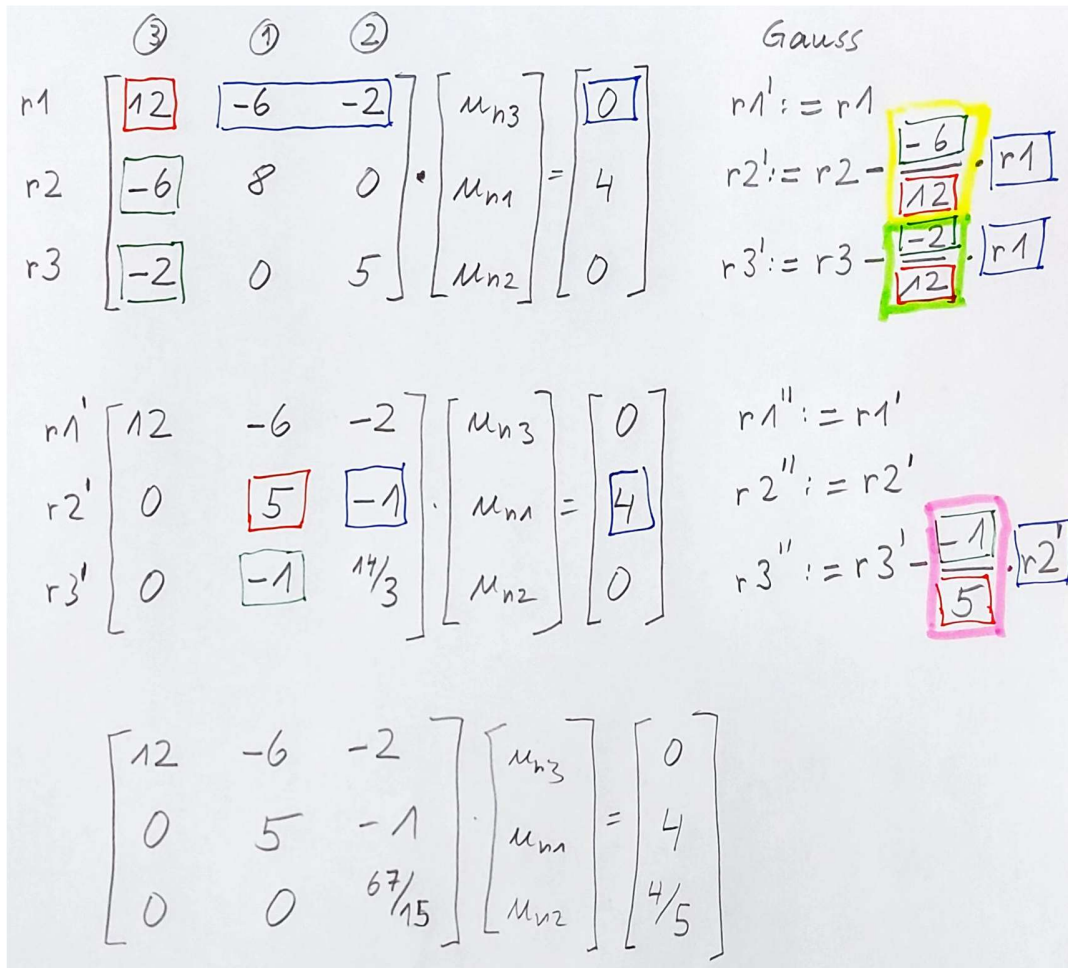
3.1 Illustrating the Gaussian elimination (LU decomposition) with an example

We set up the nodal system of the circuit in **Figure 17** and obtain the linear equation system in the first row of **Figure 18**. Please note the specific ordering of the node voltages. The first diagonal entry of 12, for instance, is the sum of admittances at node 3. Between nodes 1 and 3, we see the admittance with value 6; this results in the entries in the first row, second column, and second row, first column in the system matrix. The current source arriving at node 1 with value 4 (amplitude of the input sine/cosine at some given frequency) is entered on the right side, second position.

The Gaussian elimination comprises two steps: a triangular decomposition (*Dreieckszerlegung*) and a backward substitution (*Rücksubstitution*). **Figure 18** illustrates the triangular decomposition for the circuit example. The idea is to create zeros in the columns below the diagonal elements by a suitable linear combination of rows, which does not change the solution. The respective diagonal element is called the Pivot element (marked in red). The algorithm divides the matrix entry in the column below the Pivot element (marked in green) by the Pivot element, multiplies the Pivot row (marked in blue) with the result, and subtracts this row from the respective row. This creates the required zero in the Pivot column and changes the other values in the rows below the Pivot row. We obtain the equation system in the middle of **Figure 18**. The process is repeated with a new Pivot element. This

leads to the linear equation system at the bottom of **Figure 18** with an upper triangular form **U** of the system matrix.

Figure 18: Triangular decomposition of the system matrix of the circuit in Figure 17.



③ ① ②

$$\begin{array}{l}
 r1 \\
 r2 \\
 r3
 \end{array}
 \begin{bmatrix}
 12 & -6 & -2 \\
 -6 & 8 & 0 \\
 -2 & 0 & 5
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 u_{n3} \\
 u_{n1} \\
 u_{n2}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 4 \\
 0
 \end{bmatrix}$$

Gauss

$$\begin{array}{l}
 r1' := r1 \\
 r2' := r2 - \frac{-6}{12} \cdot r1 \\
 r3' := r3 - \frac{-2}{12} \cdot r1
 \end{array}$$

$$\begin{array}{l}
 r1' \\
 r2' \\
 r3'
 \end{array}
 \begin{bmatrix}
 12 & -6 & -2 \\
 0 & 5 & -1 \\
 0 & -1 & 14/3
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 u_{n3} \\
 u_{n1} \\
 u_{n2}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 4 \\
 0
 \end{bmatrix}$$

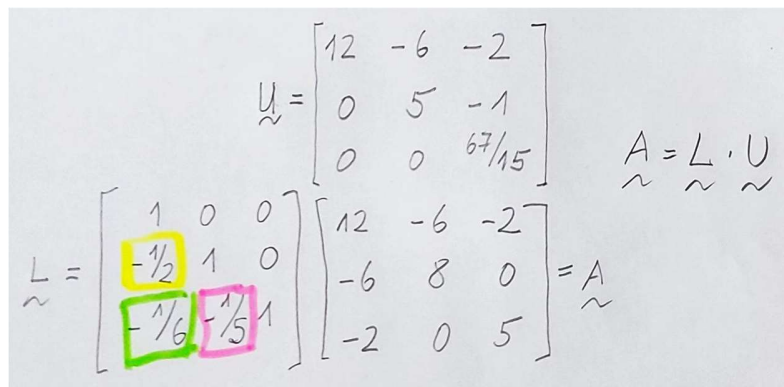
$$\begin{array}{l}
 r1'' := r1' \\
 r2'' := r2' \\
 r3'' := r3' - \frac{-1}{5} \cdot r2'
 \end{array}$$

$$\begin{bmatrix}
 12 & -6 & -2 \\
 0 & 5 & -1 \\
 0 & 0 & 67/15
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 u_{n3} \\
 u_{n1} \\
 u_{n2}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 4 \\
 4/5
 \end{bmatrix}$$

Please note that the Pivot elements must be non-zero due to the division. Swapping rows and swapping columns (this corresponds to a re-ordering of the system coordinates) is applied to achieve this.

Please note that the division elements (marked in yellow, pink, and green) can be collected in a lower triangular matrix **L** at positions corresponding to the position in the respective Pivot column and 1s as diagonal elements. The product of **L** and **U** is the original system matrix, as illustrated in **Figure 19**. This is referred to as LU decomposition.

Figure 19: LU decomposition of system matrix of Figure 18.

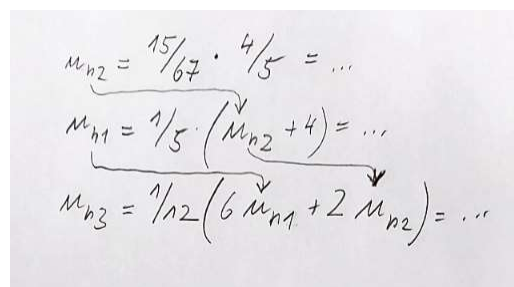


$$\tilde{L} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{6} & -\frac{1}{5} & 1 \end{bmatrix} \begin{bmatrix} 12 & -6 & -2 \\ -6 & 8 & 0 \\ -2 & 0 & 5 \end{bmatrix} = \tilde{A}$$

$$\tilde{U} = \begin{bmatrix} 12 & -6 & -2 \\ 0 & 5 & -1 \\ 0 & 0 & \frac{67}{15} \end{bmatrix} \quad \tilde{A} = \tilde{L} \cdot \tilde{U}$$

With the triangular decomposition of the linear equation system at the bottom of Figure 18, its solution can now be computed by a backward substitution. The last system coordinates results from the last row. Its value can be used in the last but one row to obtain the last but system coordinate, and so on:

Figure 20: Backward substitution of the triangular decomposition of the system matrix of the circuit in Figure 17.



$$m_{n2} = \frac{15}{67} \cdot \frac{4}{5} = \dots$$

$$m_{n1} = \frac{1}{5} \cdot (m_{n2} + 4) = \dots$$

$$m_{n3} = \frac{1}{12} (6 m_{n1} + 2 m_{n2}) = \dots$$

3.2 Gaussian elimination (LU decomposition)

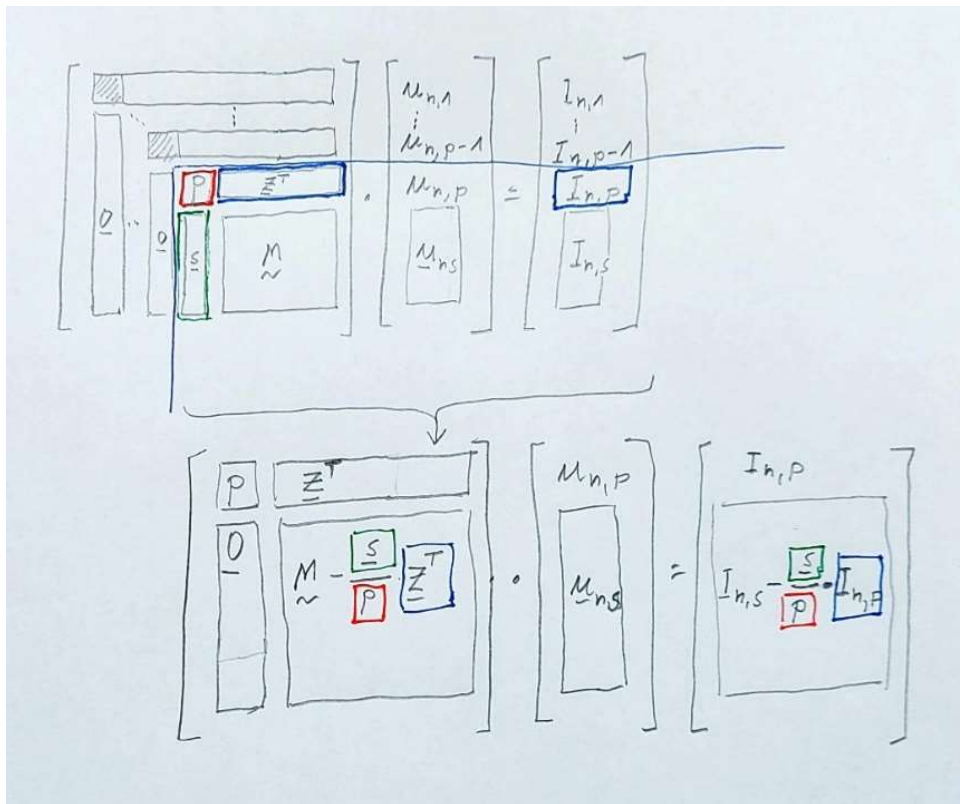
In the example in the previous section, we have made each step of a Gaussian elimination component-wise. On a more abstract level, the Gaussian elimination can be formulated as a sequence of operations on the system matrix and right-hand side that transform it into an upper triangular form:

$$\mathbf{Y}_n^{(0)} \cdot \mathbf{u}_n = \mathbf{I}_n^{(0)} \rightarrow \mathbf{Y}_n^{(1)} \cdot \mathbf{u}_n = \mathbf{I}_n^{(1)} \rightarrow \dots \rightarrow \mathbf{Y}_n^{(n-1)} \cdot \mathbf{u}_n = \mathbf{I}_n^{(n-1)}$$

$$\Leftrightarrow \mathbf{R}_n \cdot \mathbf{u}_n = \mathbf{Q}_n \quad (61)$$

Each transformation step can be formulated by multiplying the equation system with a transformation matrix. There is another way of formulating a Gaussian elimination between a component-wise formulation of the transformation and the formulation by a matrix multiplication. This intermediate formulation combines all elements in the Pivot row in a row vector \mathbf{z}^T and all elements in the Pivot column in a vector \mathbf{s} . The matrix and right-hand side elements that shall be transformed are combined in the matrix \mathbf{M} and vector $\mathbf{I}_{n,s}$, respectively. Then, the Gaussian elimination step p is formulated as shown in Figure 21.

Figure 21: A Gaussian elimination step.



3.3 Complexity of the Gaussian elimination

Inspection of the Gaussian elimination steps, as shown in the circuit example and **Figure 21**, allows us to determine the number of operations and the computational complexity of a Gaussian elimination. We define one operation as one addition or subtraction plus one multiplication or division: $1 \text{ Op} = 1 \text{ Add/Sub} + 1 \text{ Mul/Div}$.

In the first step, $n-1$ row elements of the matrix times $n-1$ column elements of the matrix must be operated. In each row, one preparatory division with the Pivot element has to be done. In the second step, this happens with $n-2$ instead of $n-1$, until one last time. This gives the overall number of operations in (62).

The preparatory division with the Pivot element can be reused for the right-hand side, and $(n-1) + (n-2) + \dots + 1$ operations have to be done. This gives the overall number of operations in (63).

For backward substitution, one operation has to be done in the first step, then two operations, and so on, till n operations in the last step. This gives the overall number of operations in (64).

Overall, the complexity of the Gaussian elimination is determined by the highest order of the number of system coordinates. That means the Gaussian elimination algorithm is of the computational order n^3 , as given in Eq. (65).

$$\mathbf{Y}_n \rightarrow \mathbf{R}_n : \sum_{i=1}^{n-1} i \cdot (i+1) = \frac{1}{3}n(n^2 - 1) \quad (62)$$

$$\mathbf{I}_n \rightarrow \mathbf{Q}_n : \sum_{i=1}^{n-1} i = \frac{1}{2}n(n-1) \quad (63)$$

$$\mathbf{R}_n \cdot \mathbf{u}_n = \mathbf{Q}_n \rightarrow \mathbf{u}_n : \sum_{i=1}^n i = \frac{1}{2}n(n+1) \quad (64)$$

$$\text{Complexity: } \mathcal{O}(n^3) \quad (65)$$

3.4 Parallel simulation approach

An approach to parallelizing the solution of a linear equation system can be imagined based on the principle of a Gaussian elimination. The requirement is a partitioning of the circuit into independent parts. We illustrate the approach with an equation system partitioned at the node set e into two subcircuits with node sets a and d without connection. Then, we reorder the node voltages such that the system matrix obtains a block structure that reflects the partitioning into two independent subcircuits. And we write the node set e two times, once for each subcircuit with temporarily different node sets b and c, i.e.,

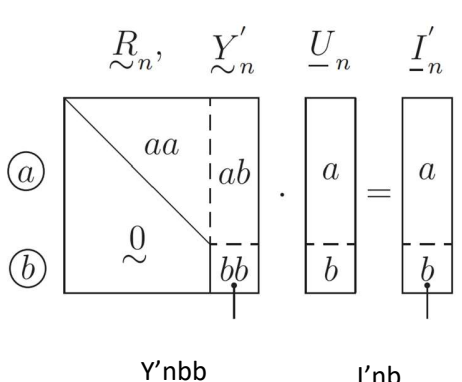
$$\mathbf{u}_{ne} = \mathbf{u}_{nb} = \mathbf{u}_{nc} \quad (66)$$

The equation system then looks like this:

$$\begin{array}{c} \textcircled{a} \\ \textcircled{b} \\ \textcircled{d} \\ \textcircled{c} \end{array} \begin{array}{|c|c|c|} \hline & \text{aa} & \text{ab} \\ \hline & \text{ba} & \text{bb} \\ \hline & \textcircled{\sim} & \\ \hline & \textcircled{\sim} & \\ \hline & & \text{dd} \\ \hline & & \text{dc} \\ \hline & & \text{cd} \\ \hline & & \text{cc} \\ \hline \end{array} \cdot \begin{array}{|c|} \hline a \\ \hline b \\ \hline d \\ \hline c \\ \hline \end{array} = \begin{array}{|c|} \hline a \\ \hline b \\ \hline d \\ \hline c \\ \hline \end{array} \quad (67)$$

As the two subcircuits with nodal voltages a+b and c+d are independent, they can be solved independently on two threads in parallel. The LU decomposition is not done completely, but for the a and d parts, in parallel:

Thread 1:

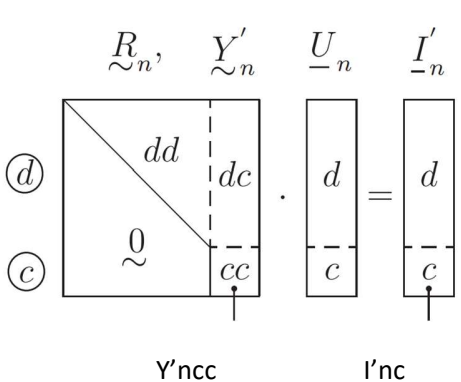


$\tilde{R}_n, \tilde{Y}'_n, \underline{U}_n, \underline{I}'_n$

(a) aa (b) $\tilde{0}$ ab bb

Y'_{nbb} I'_{nb}

Thread 2:



$\tilde{R}_n, \tilde{Y}'_n, \underline{U}_n, \underline{I}'_n$

(d) dd (c) $\tilde{0}$ dc cc

Y'_{ncc} I'_{nc}

(68)

Now, we must consider the two subcircuits' mutual influence by re-connecting them at the equal node sets b and c. According to Sec. 2.5, we obtain:

$$\begin{aligned}
 \mathbf{Y}'_{nee} &= \mathbf{Y}'_{nbb} + \mathbf{Y}'_{ncc} \quad \text{and} \quad \mathbf{I}'_{ne} = \mathbf{I}'_{nb} + \mathbf{I}'_{nc} \\
 \mathbf{Y}'_{nee} \cdot \mathbf{u}_{ne} &= \mathbf{I}'_{ne}
 \end{aligned}
 \tag{69}$$

Next, Eq. (69) is solved by a Gaussian elimination (triangular decomposition and subsequent backward elimination). Afterward, the node voltages for node-set $e = b = c$ are fed back to the threads to solve the two parts of (68) in parallel.

If the node sets a and d are large, then a large part of the Gaussian elimination of the overall system can be done in parallel. Only the solution of (69) has to be done serially.

Please note that, as linear equation systems also result in DC and TR simulations, the parallel approach holds for simulation in general.

3.4.1 Example of the parallel simulation approach

As an example, we will use the circuit in [Figure 17](#). The Gaussian elimination without partitioning leads to the equation system shown in [Figure 22](#).

Figure 22: Gaussian elimination of nodal voltage system of the circuit in Figure 17.

$$\begin{array}{l}
 \textcircled{1} \\
 \textcircled{2} \\
 \textcircled{3}
 \end{array}
 \begin{bmatrix}
 8 & 0 & -6 \\
 0 & 5 & -2 \\
 -6 & -2 & 12
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 \mu_{n1} \\
 \mu_{n2} \\
 \mu_{n3}
 \end{bmatrix}
 =
 \begin{bmatrix}
 4 \\
 0 \\
 0
 \end{bmatrix}
 \rightarrow
 \begin{bmatrix}
 8 & 0 & -6 \\
 0 & 5 & -2 \\
 0 & 0 & \frac{67}{10}
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 \mu_{n1} \\
 \mu_{n2} \\
 \mu_{n3}
 \end{bmatrix}
 =
 \begin{bmatrix}
 4 \\
 0 \\
 3
 \end{bmatrix}$$

Partitioning the circuit in [Figure 17](#) according to the red line in the figure results in the two subcircuits in [Figure 23](#). The two additional current sources are neutralizing each other in the subcircuit connection.

Figure 23: Two subcircuits after the circuit partitioning in Figure 17 at the red line.

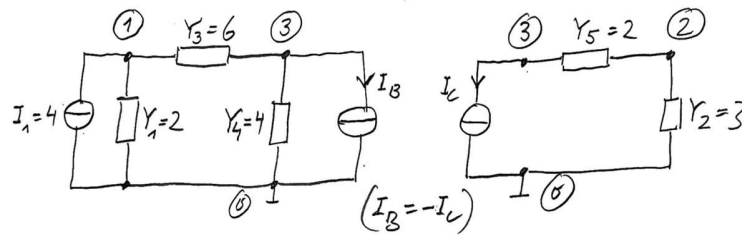


Figure 24 sketches the parallel process: the two resulting nodal voltage systems, the Gaussian elimination in the two threads, the subcircuit connection at node 3, computing the node voltage u_{n3} , and feeding the result back to the two threads.

Figure 24: Sketch of parallel simulation process according to subcircuits in Figure 23.

Thread 1:

$$\begin{matrix} \textcircled{1} & \begin{bmatrix} 8 & -6 \\ -6 & 10 \end{bmatrix} \cdot \begin{bmatrix} u_{n1} \\ u_{n3} \end{bmatrix} = \begin{bmatrix} 4 \\ -I_B \end{bmatrix} \end{matrix}$$

↓ Gauss

$$\begin{bmatrix} 8 & -6 \\ 0 & 11/2 \end{bmatrix} \cdot \begin{bmatrix} u_{n1} \\ u_{n3} \end{bmatrix} = \begin{bmatrix} 4 \\ 3 - I_B \end{bmatrix}$$

Thread 2:

$$\begin{matrix} \textcircled{2} & \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_{n2} \\ u_{n3} \end{bmatrix} = \begin{bmatrix} 0 \\ -I_C \end{bmatrix} \\ \textcircled{3} & \end{matrix}$$

↓ Gauss

$$\begin{bmatrix} 5 & -2 \\ 0 & 6/5 \end{bmatrix} \cdot \begin{bmatrix} u_{n2} \\ u_{n3} \end{bmatrix} = \begin{bmatrix} 0 \\ -I_C \end{bmatrix}$$

↙ connecting node 3 ↘

$$\left[\frac{11}{2} + \frac{6}{5} \right] \cdot u_{n3} = 3 - I_B - I_C$$

→ u_{n3}

↙ Thread 1 → u_{n1} ↘ Thread 2 → u_{n2}

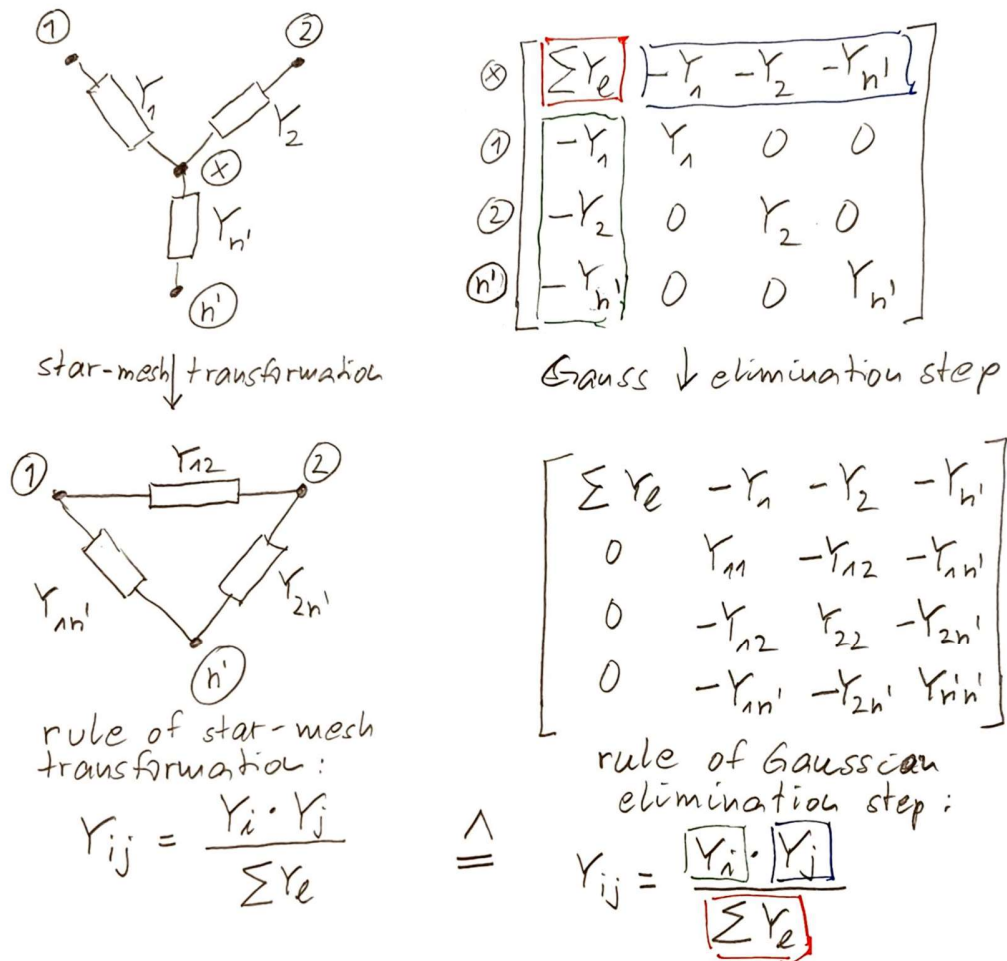
3.5 Interpreting a Gaussian elimination step by circuitry

The star-mesh transformation with three edges is illustrated on the left side of Figure 25. The node x is eliminated. The new admittances are calculated with the formula at the bottom of the left side. This formula comes from applying Kirchhoff and Ohm laws at terminal nodes.

The top right side of Figure 25 gives the nodal admittance corresponding to the star circuit on the left. Below, the first Gaussian elimination step is made. The Gaussian elimination rule to compute the new values Y_{ij} is given at

the bottom. We can see that it matches the rule of the star-mesh transformation for all $i \neq j$. But what about a Y_{ii} , which does not arise in the star-mesh formula?

Figure 25: Star-mesh transformation and a Gaussian elimination step ("system reduction").



According to the Gaussian elimination step, we obtain for a diagonal element in the system matrix of the star circuit:

$$Y_{ii} = Y_i - \frac{Y_i^2}{\sum Y_l} \tag{70}$$

Two mathematical transformations lead to the following equation.

$$Y_{ii} = \frac{Y_i \sum Y_l - Y_i^2}{\sum Y_l} = \frac{\sum_{l \neq i} Y_i Y_l}{\sum Y_l} \tag{71}$$

Looking again at the rule of Gaussian elimination in Figure 25 shows that (71) describes the sum of all non-diagonal elements in the respective row i of the system matrix after the first elimination step:

$$Y_{ii} = \sum_{j \neq i} Y_{ij} \quad (72)$$

This matches what we see in the mesh circuit: at each node i , all admittances with $j \neq i$ arrive.

Overall, we can see that the star-mesh transformation is a circuit-equivalent interpretation of a Gaussian elimination step.

3.6 Structural interpretation of a Gaussian elimination step

In the following, we will look at the nodal admittance matrix from a structural point of view. Specifically, we will abstract it into zero-elements and non-zero elements. The structure matrix \mathbf{S} to a nodal admittance matrix \mathbf{Y}_n will have 0 as an entry when the corresponding admittance is zero and 1 when the corresponding admittance is non-zero:

$$S_{ij} = \begin{cases} 0, & Y_{nij} = 0 \\ 1, & Y_{nij} \neq 0 \end{cases} \quad (73)$$

We will assume that

- the structure matrix is symmetric and
- the diagonal elements of the structure matrix are all 1.

An undirected structure graph $G=(V,E)$ corresponds to the structure matrix with the vertices V and the edges E :

$$E = \{[i, j] | i, j \in V, i \neq j, S_{ij} = 1\}, \quad [i, j] = \{(i, j), (j, i)\} \quad (74)$$

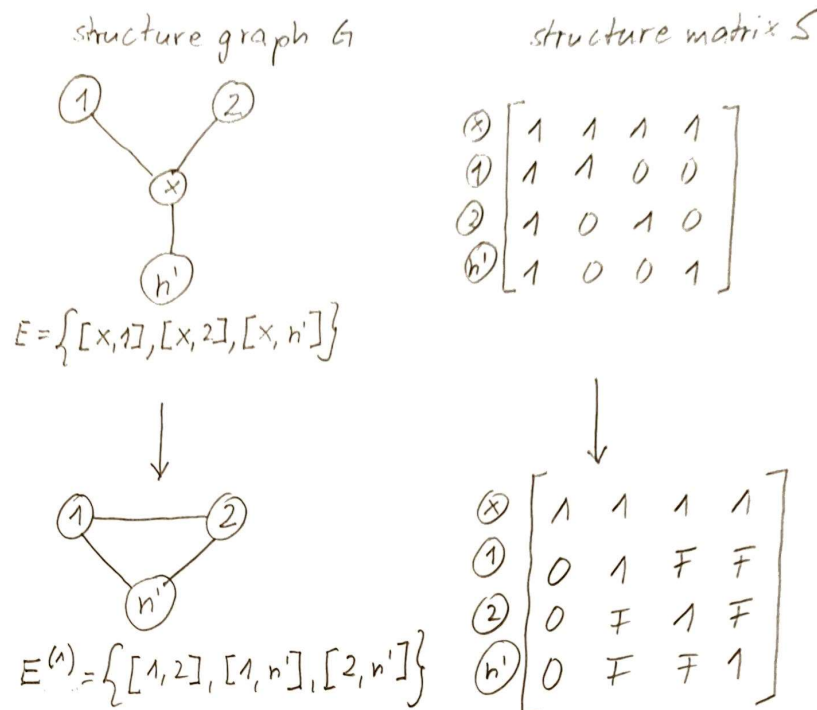
The star-mesh transformation/Gaussian elimination example then has the form shown in **Figure 26**. With an F , we denote a so-called fill-in. This is a zero-entry in the structure matrix that becomes a non-zero entry after the Gaussian elimination step. It corresponds to a new edge in the structure graph that did not exist before the Gaussian elimination step.

From the visual inspection of **Figure 26**, we can identify the following condition for creating fill-ins:

Rule: Each pair of edges $[k,i]$ and $[k,j]$ in the structure graph G leads to an edge $[i,j]$ after the elimination of vertex/node k . If this edge did not exist in G before the elimination of k , it is newly created, which refers to 2 fill-ins in the structure matrix \mathbf{S} (because of an undirected edge). (75)

In other words, after eliminating a vertex/node k , all vertices connected to k form a complete graph, i.e., a fully meshed graph, i.e., each pair of vertices has an edge. In an informal way, you can imagine to pull, e.g., node x in **Figure 26** out of the page, then cut all its connections, and connect all nodes with each other that were neighbors of node x .

Figure 26: Structure graph/matrix of the Star-mesh transformation/Gaussian elimination step ("system reduction").



3.7 Computational cost of a Gaussian elimination step considering zero-elements

Let us take a look at a Gaussian elimination step with consideration of zero-elements in the system. That means:

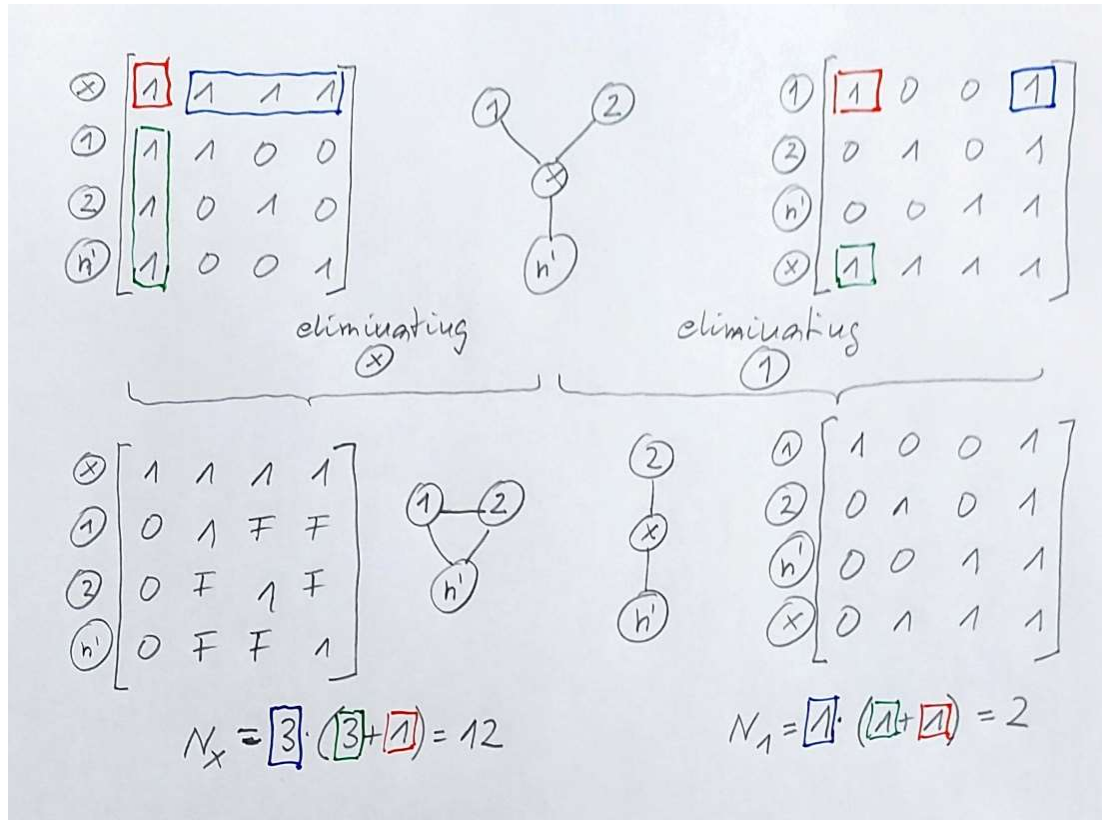
- Only non-zero elements are stored.
- No operations are made on zero-elements.

Figure 27 shows the structure graphs and structure matrices of the star-mesh transformation under consideration of zero-elements for two cases: once, node x is eliminated (left), and once, node 1 (right). This refers to different orderings of the Pivots and the corresponding ordering of the matrix operations.

The colors indicate the essential Gaussian elimination operation defined in **Figure 18** and **Figure 21**. We can see that eliminating node x implies three non-zero elements in the Pivot row and Pivot column, which corresponds to the structure graph, where node x is connected to the other three nodes. The elimination requires operations on all elements of the submatrix of nodes $1, 2, h'$. Previous zero-elements become non-zeros, leading to 6 fill-ins.

On the other hand, eliminating node 1 , which is only connected to node x in the structure graph, requires only operations on one element in the structure matrix. In addition, it does not create any fill-in.

Figure 27: Structure graph/matrix of the Star-mesh transformation/Gaussian elimination step when eliminating node x (left) and node 1 (right).



We define the degree $g(k)$ of a node k in the structure graph G :

Degree of node k : $g(k)$ = number of neighbor nodes of node k in the structure graph G ,
i.e., nodes connected to node k (76)

The degrees of nodes x and 1 are: $g(x)=3$, $g(1)=1$. From the basic operation in a Gaussian elimination, we can then formulate the Gaussian elimination cost considering zero-elements:

$$\text{Computational cost of Gaussian elimination step considering zero-elements } N_k = g(k) \cdot [g(k) + 1] \quad (77)$$

The computational cost N_k is analogous to the inner part in the sum of Eq. (62) with the difference that operations are not done on all columns and rows but only those N_k columns/rows with non-zero elements.

The goal is to save computational cost and maintain the sparsity of the system equations. This is achieved in two ways:

- Consider zero-elements in the equations (i.e., use sparse matrix methods). This is illustrated in Sec. 3.8.
- Determine a heuristic for Pivoting that maintains the sparsity. This is illustrated in Sec. 3.8.1.

3.8 Structural Gaussian elimination considering zero-elements

In the following, we will perform an exemplary Gaussian elimination on the structure graph under consideration of zero-elements. We will first apply the natural Pivoting order, i.e., in the arbitrarily given order. Afterward, we will describe a suboptimal heuristic for Pivoting to maintain sparsity. This will then be applied to the same example. We will compare the differences between not considering zero-elements, considering zero-elements with natural Pivoting, and considering zero-elements with suboptimal heuristic Pivoting.

Figure 28 shows the structural elimination process with natural Pivoting for the structure graph example in the last column at the top. Four columns are shown. The first column gives the node that will be eliminated. The second column gives the computational cost to eliminate this node according to Eqs. (76) and (77). The third column shows the number of fill-ins after eliminating the respective node according to rule (75). The last row gives the overall computational cost and overall number of fill-ins.

The computational cost without consideration of zero-elements with $n=8$ nodes would have been, according to Eq. (62):

$$N_0 = \frac{1}{3}n \cdot (n^2 - 1) = \frac{1}{3}8 \cdot (8^2 - 1) = 168 \quad (78)$$

This yields a computational saving CS obtained by the consideration of zero-elements of:

$$CS = \frac{168 - 88}{168} = 47.6\% \quad (79)$$

Now, we present a simple heuristic to create a suboptimal Pivot ordering to save computational costs further by reducing the number of fill-ins. It is a heuristic, as it is based on plausibility. It is suboptimal because it cannot be proven to lead to the optimal result.

3.8.1 Suboptimal Pivoting Rules

- 1) Eliminate nodes of degree 1.
- 2) Eliminate a node with a minimum number of fill-ins.
- 3) In the case that criterion 2) provides alternatives, select a node with the highest degree.

Rule 1 refers to nodes with a minimum computational cost of $N_k=2$ and that do not create any fill-in, as their degree is 1. These are “appendix” nodes and should be eliminated first.

Rule 2 requires some effort, as the number of fill-ins for all possible nodes has to be computed. In practice, a simplification of this rule, e.g., the one in Sec. 3.8.2, is often used.

Rule 3 is motivated by the idea that the early elimination of nodes with many connections to other nodes will reduce the probability of creating fill-ins later in the elimination process: the more edges disappear, the fewer fill-ins probably later.

Figure 28: Structural Gaussian elimination example considering zero-elements – natural Pivoting.

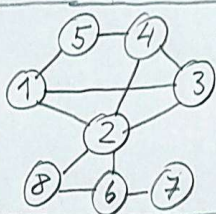
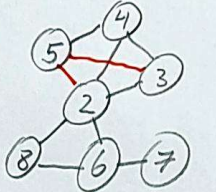
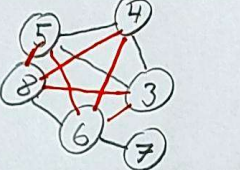
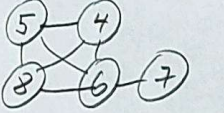
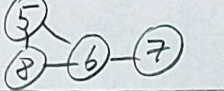
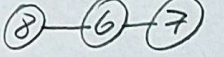

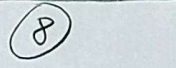
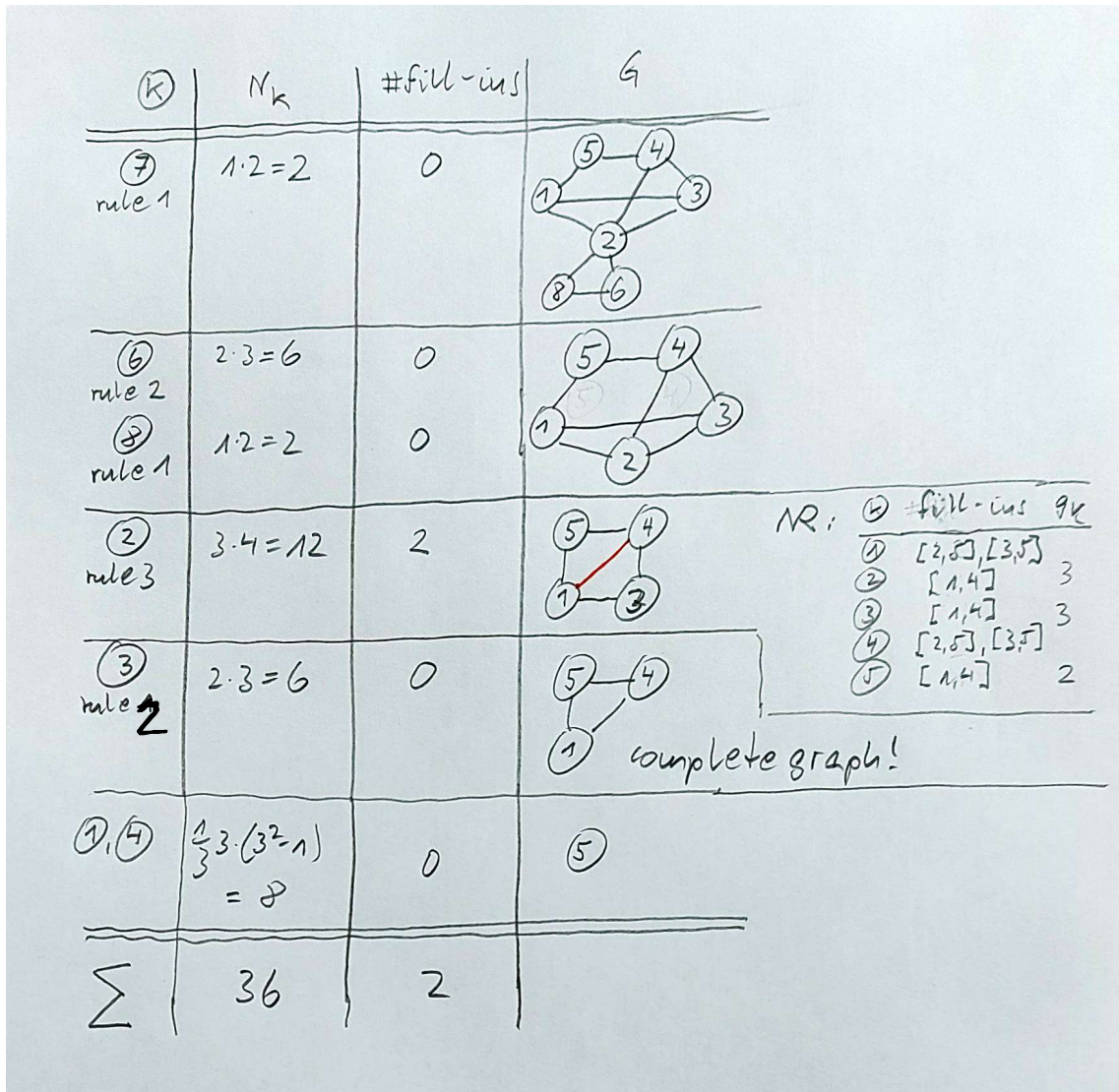
k	N_k	# fill-ins	G
			
①	$3 \cdot 4 = 12$	4	
②	$5 \cdot 6 = 30$	12	
③	$4 \cdot 5 = 20$	0	
④	$3 \cdot 4 = 12$	0	
⑤	$2 \cdot 3 = 6$	0	
⑥	$2 \cdot 3 = 6$	2	
⑦	$1 \cdot 2 = 2$	0	
Σ	88	18	

Figure 29 shows the structural elimination process of the example from Figure 28 with suboptimal Pivoting. We can see much fewer fill-ins and a much smaller overall computational effort.

Figure 29: Structural Gaussian elimination example from Figure 28 considering zero-elements – suboptimal Pivoting.



The computational saving compared to not considering zero-elements amounts to:

$$CS = \frac{168 - 36}{168} = 78.6\% \quad (80)$$

3.8.2 Markovitz Pivoting

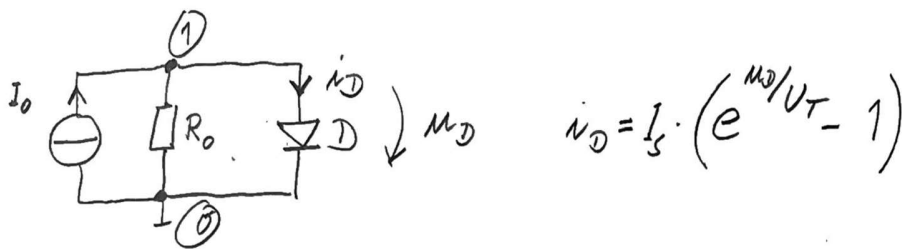
The Pivoting rules according to 3.8.1 bring much effort that might consume the savings later on. A straightforward alternative for Pivoting is to order along the computational cost, i.e., the minimum degree of nodes, according to Eq. (76) and (77). This cost is called Markovitz product MP.

$MP = [g(k)]^2$ in the case of a symmetric matrix; $MP = \text{\#non-zeros in a Pivot column} \times \text{\#non-zeros in a Pivot row}$ in the case of an asymmetric matrix.

4. DC analysis

A DC analysis (*Gleichstrom-Analyse*) analyzes the stationary circuit behavior on a constant input. The result is called the DC operating point. A DC analysis is done before an AC or TR analysis. The main mathematical task of a DC analysis is dealing with the nonlinear elements in the circuit, which means the solution of nonlinear algebraic equations or, more simply, root finding. **Figure 30** illustrates an exemplary nonlinear circuit with a diode.¹

Figure 30: Circuit example with a nonlinear element.



Kirchhoff current law at node 1 leads to the following equation:

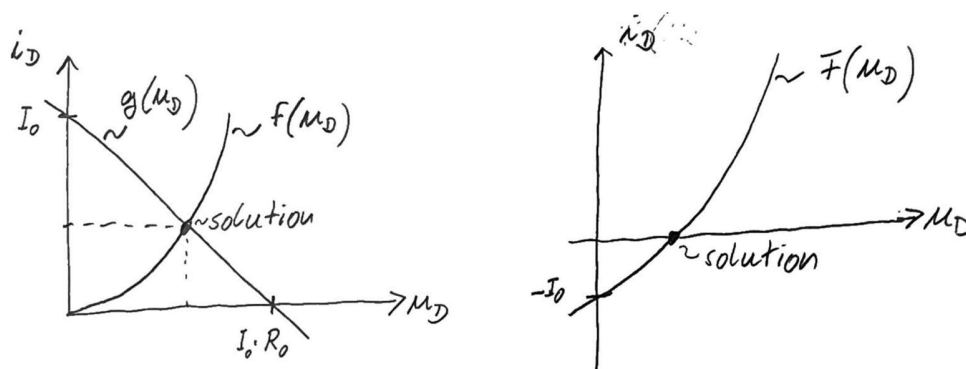
$$i_D + \frac{u_D}{R_0} - I_0 = 0 \quad (81)$$

Eq. (81) can be partitioned into a nonlinear part f and a linear part g :

$$\underbrace{I_s \left(\exp \frac{u_D}{U_T} - 1 \right)}_{f(u_D)} = \underbrace{I_0 - \frac{u_D}{R_0}}_{g(u_D)} \Leftrightarrow f(u_D) - g(u_D) = F(u_D) = 0 \quad (82)$$

The root finding for this one-dimensional problem (82) can be visualized in the two ways shown in **Figure 31**.

Figure 31: Root finding of the example in Figure 30.



¹ Exponential behavior can be found in a transistor's subthreshold region.

On the left, the intersection point of the linear part g and the nonlinear part f of (82) is the solution. On the right, the root of the overall function F is the solution.

In the following two sections, the one-dimensional case $F(x)=0$ of the general task of root finding,

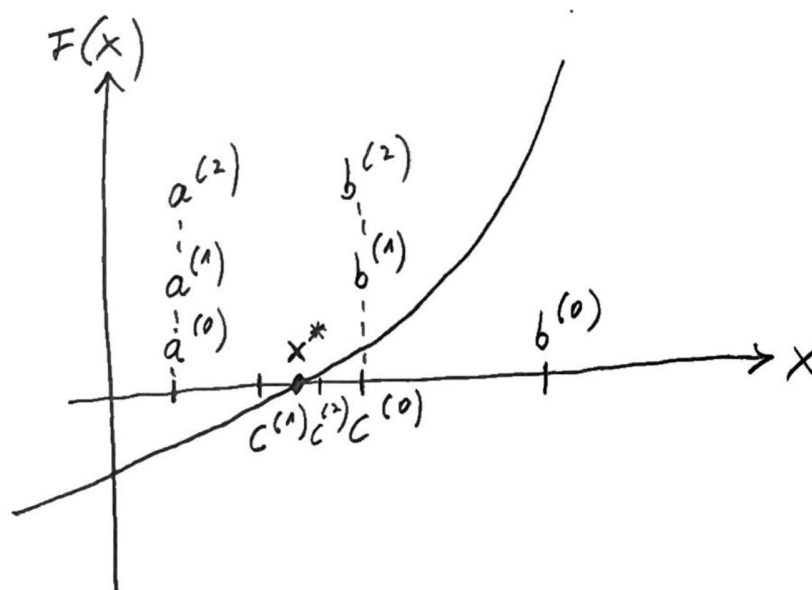
$$F(x) = 0 \quad (83)$$

will be considered.

4.1 Bi-sectioning (one-dimensional) for root finding

Bi-sectioning starts from an interval that includes the/one single root and assumes a monotonous behavior within the interval. That means the signs of the left and right interval bounds are opposite. The center of the interval is determined. The center becomes the new left or right interval bound such that the signs of the new left and right interval bounds stay opposite. This guarantees the root is within the new interval of half the previous width. This proceeding is iteratively repeated until the interval width corresponds to the required accuracy of the root. **Figure 32** illustrates two iteration steps of the bi-sectioning process.

Figure 32: Graphical illustration of one-dimensional bi-sectioning.



An incomplete pseudo code for the procedure could look like this:

- 1) Determine start interval $[a,b]$ with $a < x^* < b$
- 2) Compute interval center $c := (a+b)/2$
- 3) If $\text{sign}[F(c)] == \text{sign}[F(b)]$ then $b := c$
- 4) If $\text{sign}[F(c)] == \text{sign}[F(a)]$ then $a := c$
- 5) Repeat from step 2) onward until $|a-b| < \Delta^*$

4.1.1 Convergence rate of bi-sectioning

The general formula to determine the convergence rate is the following:

$$|\varepsilon^{(\mu+1)}| \leq L \cdot |\varepsilon^{(\mu)}|^p \quad \text{with } \varepsilon = x - x^* \text{ and } |\varepsilon| < 1 \quad (84)$$

L is the convergence factor, and p describes the convergence rate as follows:

$$\begin{aligned} \text{linear convergence:} & \quad p = 1 \wedge 0 < L < 1 \\ \text{quadratic convergence:} & \quad p = 2 \end{aligned} \quad (85)$$

Eq. (84) requires that the error ε , i.e., the difference between the intermediate solution in the current iteration step μ and the needed solution, has an absolute value below 1 to reduce the error in each iteration step. Linear convergence is more slowly than quadratic convergence. Quadratic convergence gives 1, 2, 4, 16, ... bits of accuracy in each iteration step; linear convergence reduces the error only by a particular factor. The convergence rate of an algorithm may be between linear and quadratic. This is called superlinear convergence and can be described by a non-integer p with $1 < p < 2$.

The error of the bi-sectioning method is defined by the interval width $\Delta = |b-a|$. The interval width is reduced by 0.5 in each iteration step. This means a linear convergence $p=1$ with a convergence factor of $L=1/2$. We can determine the error in the μ -th iteration step in dependence on the error at the start:

$$\varepsilon^{(\mu)} = \frac{1}{2} \varepsilon^{(\mu-1)} = \left(\frac{1}{2}\right)^{(\mu)} \cdot \varepsilon^{(0)} \quad (86)$$

Solving (86) for μ gives the number of iteration steps required to get to a defined accuracy from the start interval:

$$\mu = \frac{1}{\log 2} \cdot \log \frac{\varepsilon^{(0)}}{\varepsilon^{(\mu)}} \quad (87)$$

A method with quadratic convergence is the Newton-Raphson method, which will be discussed in the following.

4.2 Newton-Raphson method for root finding (one-dimensional)

The Newton-Raphson method is based on linearization, i.e., Taylor series with first-order derivative, of the nonlinear function $F(x)$:

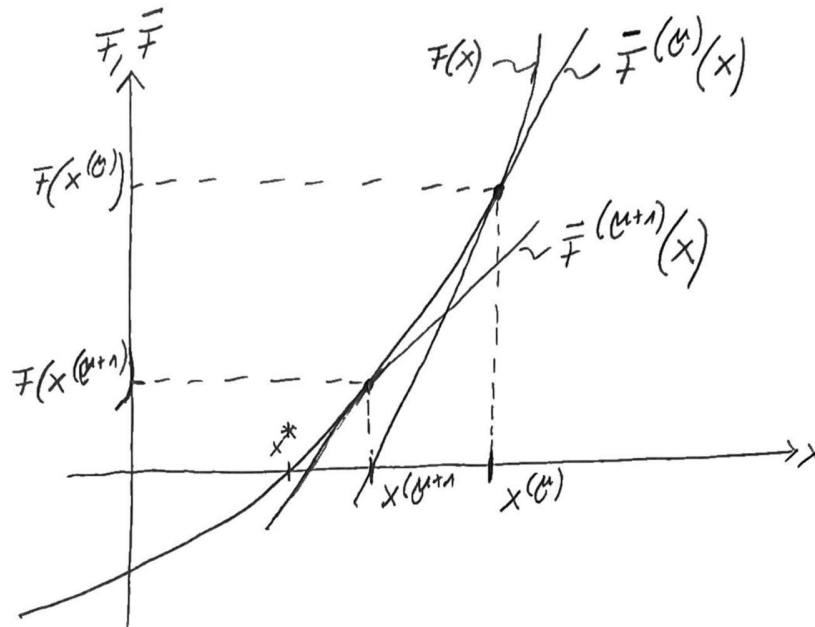
$$\bar{F}^{(\mu)}(x) = F(x^{(\mu)}) + F'(x^{(\mu)}) \cdot (x - x^{(\mu)}) \quad (88)$$

Then, we set \bar{F} instead of F equal to 0, resolve the formula for x, and take this x as the next iterative solution for x:

$$\bar{F}^{(\mu)}(x) = 0 \Leftrightarrow x^{(\mu+1)} = x^{(\mu)} - \frac{F(x^{(\mu)})}{F'(x^{(\mu)})} \quad (89)$$

Figure 33 illustrates the Newton-Raphson approach. It shows the nonlinear function $F(x)$ whose root x^* must be calculated. At the iteration point $x^{(\mu)}$, the linearization $\bar{F}^{(\mu)}$ is shown. It is the tangent to $F(x)$. The new iteration point $x^{(\mu+1)}$ is where $\bar{F}^{(\mu)}$ has its root. Then, the process is repeated with a new linearization $\bar{F}^{(\mu+1)}$ at point $x^{(\mu+1)}$.

Figure 33: Graphical illustration of one-dimensional Newton-Raphson method for root finding.



4.2.1 Convergence rate of Newton-Raphson

We expand the functions F and F' in (89) separately in their Taylor series at iteration point $x^{(\mu)}$ around the solution x^* up to the order n . This results in the following equation:

$$x^{(\mu+1)} = x^{(\mu)} - \frac{F(x^*) + F'(x^*) \cdot \overbrace{(x^{(\mu)} - x^*)}^{\varepsilon^{(\mu)}} + \frac{1}{2}F''(x^*) \cdot \varepsilon^{(\mu)2} + \dots + \frac{1}{n!}F^{(n)}(x^*) \cdot \varepsilon^{(\mu)n}}{F'(x^*) + F''(x^*) \cdot \varepsilon^{(\mu)} + \dots + \frac{1}{(n-1)!}F^{(n)}(x^*) \cdot \varepsilon^{(\mu)^{n-1}}} \quad (90)$$

We subtract x^* from $x^{(\mu+1)}$ on the left side and from $x^{(\mu)}$ on the right side and include that $F(x^*)=0$ to obtain:

$$\varepsilon^{(\mu+1)} = \varepsilon^{(\mu)} - \varepsilon^{(\mu)} \cdot \frac{F'(x^*) + \frac{1}{2}F''(x^*) \cdot \varepsilon^{(\mu)} + \dots + \frac{1}{n!}F^{(n)}(x^*) \cdot \varepsilon^{(\mu)^{n-1}}}{F'(x^*) + F''(x^*) \cdot \varepsilon^{(\mu)} + \dots + \frac{1}{(n-1)!}F^{(n)}(x^*) \cdot \varepsilon^{(\mu)^{n-1}}} \quad (91)$$

We will now distinguish the two cases of a single root and a root of order n .

Single root: $F'(x^*) \neq 0$

In the case of a single root, we drop all terms in (91) higher than the second-order derivative F'' . This results in (92). Using a formula on a series expansion that holds for $\varepsilon \ll 1$, we can approximate the fraction in (92) as given in (93).

Dropping the quadratic term in ε after multiplying the inner brackets leads to (94) and (95).

$$\varepsilon^{(\mu+1)} \approx \varepsilon^{(\mu)} \left(1 - \frac{1 + \frac{1}{2}\varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)}}{1 + \varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)}} \right) \quad (92)$$

$$\approx \varepsilon^{(\mu)} \left(1 - \left(1 + \frac{1}{2}\varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)} \right) \left(1 - \varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)} \right) \right) \quad (93)$$

$$\approx \varepsilon^{(\mu)} \left(1 - \left(1 + \frac{1}{2}\varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)} - \varepsilon^{(\mu)} \frac{F''(x^*)}{F'(x^*)} \right) \right) \quad (94)$$

$$= \frac{1}{2} \frac{F''(x^*)}{F'(x^*)} \varepsilon^{(\mu)2} \quad (95)$$

According to (95), the Newton-Raphson approach has quadratic convergence and a convergence factor of $1/2 \cdot F''/F'$ in the case of a single root.

We will illustrate this with a numerical example:

$$F(x) = e^x - 1 \quad \text{solution: } x^* = 0, \quad F(0) = 0 \quad (96)$$

The first and second-order derivatives of F are:

$$F'(x) = e^x = F''(x), \quad F'(0) = F''(0) = 1 \quad (97)$$

Inserting the result of (97) into (95) gives the following convergence rate:

$$\varepsilon^{(\mu+1)} \approx \frac{1}{2} \varepsilon^{(\mu)2} \quad (98)$$

Now, we validate (98) by setting up the Newton-Raphson iteration function according to (89),

$$x^{(\mu+1)} = x^{(\mu)} - \frac{e^{x^{(\mu)}} - 1}{e^{x^{(\mu)}}} \quad (99)$$

and performing some iteration steps starting from $x^{(0)}=1$:

μ	0	1	2	3
$x^{(\mu)}$	1.0000	0.3679	0.0601	0.0018

Knowing that the solution is $x^*=0$, steps 2 and 3 match (98).

Root of order n: $F'(x^*) = \dots = F^{(n-1)}(x^*) = 0$ $F^{(n)}(x^*) \neq 0$

In the case of a root of order n, only the last terms in the nominator and denominator of (91) remain. This leads to:

$$\varepsilon^{(\mu+1)} \approx \varepsilon^{(\mu)} \left(1 - \frac{(n-1)!}{n!} \right) = \varepsilon^{(\mu)} \left(1 - \frac{1}{n} \right) = \frac{n-1}{n} \varepsilon^{(\mu)} \quad (100)$$

According to (100), the Newton-Raphson approach has linear convergence and a convergence factor of (n-1)/n in the case of a root of order n.

An illustration will be done with the following numerical example:

$$F(x) = x^3 \quad \text{solution: } x^* = 0, \quad F(0) = 0 \quad (101)$$

We obtain for the first three derivatives of F:

$$F'(x) = 3x^2, \quad F''(x) = 6x, \quad F'''(x) = 6, \quad F'(0) = F''(0) = 0 \quad (102)$$

This shows that F has a root of order 3. Inserting the result of (102) into (100) gives the following convergence rate:

$$\varepsilon^{(\mu+1)} \approx \frac{n-1}{n} \varepsilon^{(\mu)} = \frac{2}{3} \varepsilon^{(\mu)} \quad (103)$$

Now, we set up the Newton-Raphson iteration function according to (89):

$$x^{(\mu+1)} = x^{(\mu)} - \frac{x^{(\mu)3}}{3x^{(\mu)2}} = \frac{2}{3} x^{(\mu)} \quad (104)$$

Eq. (104) matches (103), considering that $x^*=0$, which validates (100).

4.3 Convergence rate of an arbitrary iteration function for root finding (one-dimensional)

Let us assume any iteration function for root finding:

$$x^{(\mu+1)} = \phi(x^{(\mu)}) \quad \text{fix point: } x^* = \phi(x^*) \quad (105)$$

Expanding ϕ around x^* yields:

$$x^{(\mu+1)} = \underbrace{\phi(x^*)}_{x^*} + \phi'(x^*) \cdot (x^{(\mu)} - x^*) + \frac{1}{2} \phi''(x^*) \cdot (x^{(\mu)} - x^*)^2 + \dots \quad (106)$$

Inserting x^* for $\phi(x^*)$, and bringing it to the left side, and considering the definition of ε results in:

$$\varepsilon^{(\mu+1)} = \phi'(x^*) \cdot \varepsilon^{(\mu)} + \frac{1}{2}\phi''(x^*) \cdot \varepsilon^{(\mu)2} + \dots \quad (107)$$

The conditions for linear convergence and quadratic convergence can be formulated based on (107):

$$\begin{aligned} \phi'(x^*) \neq 0 \wedge |\phi'(x^*)| < 1 &\rightarrow \text{linear convergence} \\ \phi'(x^*) = 0 \wedge \phi''(x^*) \neq 0 &\rightarrow \text{quadratic convergence} \end{aligned} \quad (108)$$

As we require $|\varepsilon| < 1$, the term ε dominates ε^2 : If $\phi' \neq 0$, ε will cover ε^2 , hence linear convergence.

Numerical example 1:

Iteration function ϕ , starting point $x^{(0)}$ and solution x^* are:

$$x^{(\mu+1)} = \phi(x^{(\mu)}) = x^{(\mu)2}; \quad x^{(0)} = c, 0 < c < 1; \quad x^* = 0 \quad (109)$$

The first and second-order derivatives of ϕ are as follows, which yields the convergence rate:

$$\begin{aligned} \phi'(x) = 2x, \quad \phi''(x) = 2 \\ \phi'(0) = 0 \wedge \phi''(0) \neq 0 \end{aligned} \rightarrow \text{quadratic convergence} \quad (110)$$

Numerical example 2:

Iteration function ϕ , starting point $x^{(0)}$ and solution x^* are:

$$x^{(\mu+1)} = \phi(x^{(\mu)}) = \sqrt{x^{(\mu)}}; \quad x^{(0)} = c, c > 0; \quad x^* = 1 \quad (111)$$

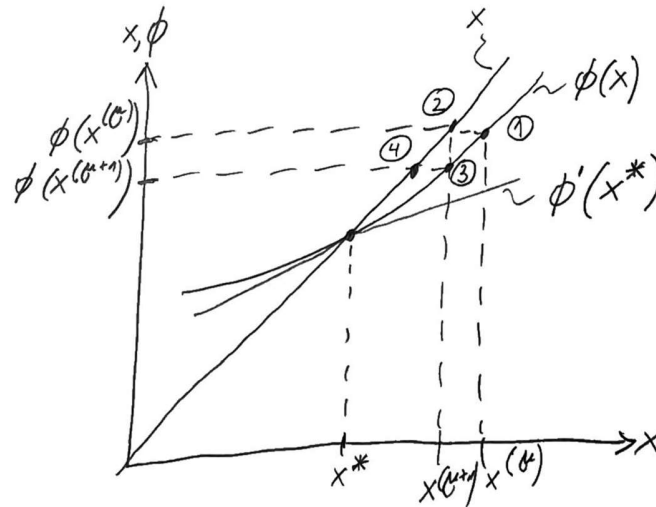
The first-order derivative of ϕ is as follows, which yields the convergence rate:

$$\phi'(x) = \frac{1}{2}x^{-\frac{1}{2}} \rightarrow \phi'(1) = \frac{1}{2} \neq 0 \wedge |\phi'(1)| < 1 \rightarrow \text{linear convergence} \quad (112)$$

The process of an arbitrary iteration function for root finding can be visualized, as shown in **Figure 34**. It shows two curves, the iteration function ϕ and the function $y=x$. The current iteration step μ corresponds to the point marked with ①. Moving horizontally from ① to the y -axis gives $\phi(x^{(\mu)})$. We go on this horizontal line “back” to point ② on the function $y=x$. We are going vertically down to the x -axis from point ②, this results in $x^{(\mu+1)}$ because of $y=x$ and the correspondence to $x^{(\mu+1)} = \phi(x^{(\mu)})$. After going “back up” to point ③, the whole process is repeated.

Figure 34 illustrates the situation for $|\phi(x^*)| < 1$. You can try a function ϕ with $|\phi(x^*)| > 1$ and find out how it does not converge.

Figure 34: Graphical illustration of one-dimensional iteration function ϕ for root finding.



4.4 Multivariate Newton-Raphson iteration function for root finding

In the following, we will formulate the Newton-Raphson iteration in the general multivariate case and analyze the convergence property in the multivariate way. The function \mathbf{F} and the coordinate \mathbf{x} now are vectors with r components:

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \mathbf{F} = [F_1 \dots F_r]^T, \mathbf{x} = [x_1 \dots x_r]^T \quad (113)$$

The linearization becomes a matrix-vector formulation:

$$\bar{\mathbf{F}}^{(\mu)}(\mathbf{x}^{(\mu+1)}) = \mathbf{F}(\mathbf{x}^{(\mu)}) + \mathbf{J}(\mathbf{x}^{(\mu)}) \cdot (\mathbf{x}^{(\mu+1)} - \mathbf{x}^{(\mu)}) = \mathbf{0} \quad (114)$$

\mathbf{J} is denoted as the Jacobian matrix, functional matrix, or sometimes sensitivity matrix. It comprises the partial derivatives of all individual functions F_i with regard to all individual coordinates x_i in the following way:

$$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}^T} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial x_1} & \dots & \frac{\partial \mathbf{F}}{\partial x_r} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial \mathbf{x}^T} \\ \vdots \\ \frac{\partial F_r}{\partial \mathbf{x}^T} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \dots & \frac{\partial F_1}{\partial x_r} \\ \vdots & & \vdots \\ \frac{\partial F_r}{\partial x_1} & \dots & \frac{\partial F_r}{\partial x_r} \end{bmatrix} \quad (115)$$

\mathbf{J} is a quadratic, nonsingular matrix. This provides a unique solution of (113), i.e., unique node voltages in the simulation. (115) provides notations of \mathbf{J} component-wise, by row vectors, and by column vectors. Eq. (114) is reformulated as a linear equation system whose solution provides the next iteration point $\mathbf{x}^{(\mu+1)}$:

$$\mathbf{J}(\mathbf{x}^{(\mu)}) \cdot \mathbf{x}^{(\mu+1)} = \mathbf{J}(\mathbf{x}^{(\mu)}) \cdot \mathbf{x}^{(\mu)} - \mathbf{F}(\mathbf{x}^{(\mu)}) \rightarrow \mathbf{x}^{(\mu+1)} \quad (116)$$

4.4.1 Convergence rate of Newton-Raphson

We will analyze the convergence property of the Newton-Raphson iteration based on the iteration function ϕ following the approach in Sec. 4.3. We obtain the multivariate iteration function ϕ of the Newton-Raphson iteration from (116):

$$\mathbf{x}^{(\mu+1)} = \phi(\mathbf{x}^{(\mu)}) = \mathbf{x}^{(\mu)} - \mathbf{J}^{-1}(\mathbf{x}^{(\mu)}) \cdot \mathbf{F}(\mathbf{x}^{(\mu)}) \quad (117)$$

As in Sec. 4.3, the iteration function is developed into a Taylor series. We will not compute the second-order term but abstract it as a term of second order $\mathcal{O}\{\cdot\}$. This leads to:

$$\mathbf{x}^{(\mu+1)} = \underbrace{\phi(\mathbf{x}^*)}_{\mathbf{x}^*} + \left. \frac{\partial \phi}{\partial \mathbf{x}^T} \right|_{\mathbf{x}^*} \cdot (\mathbf{x}^{(\mu)} - \mathbf{x}^*) + \mathcal{O}\{(\mathbf{x}^{(\mu)} - \mathbf{x}^*) \cdot (\mathbf{x}^{(\mu)} - \mathbf{x}^*)^T\} \quad (118)$$

Reformulating (118) in dependence of the difference between an \mathbf{x} and the solution \mathbf{x}^* leads to:

$$\boldsymbol{\varepsilon}^{(\mu+1)} = \left. \frac{\partial \phi}{\partial \mathbf{x}^T} \right|_{\mathbf{x}^*} \cdot \boldsymbol{\varepsilon}^{(\mu)} + \mathcal{O}\{\boldsymbol{\varepsilon}^{(\mu)} \cdot \boldsymbol{\varepsilon}^{(\mu)T}\} \quad (119)$$

To calculate the first-order derivative $\partial \phi / \partial \mathbf{x}^T$, we reformulate the matrix-vector product in the iteration function (117):

$$\phi(\mathbf{x}) = \mathbf{x} - \mathbf{J}^{-1} \cdot \mathbf{F} = \mathbf{x} - [\mathbf{a}_1 \dots \mathbf{a}_r] \cdot \begin{bmatrix} F_1 \\ \vdots \\ F_r \end{bmatrix} = \mathbf{x} - \sum_{i=1}^r \mathbf{a}_i \cdot F_i \quad (120)$$

Eq. (120) has used the property that a matrix-vector product can be reformulated as a linear combination of the columns of the matrix, where each column is weighted with the corresponding component of the vector. We can now calculate the derivative $\partial \phi / \partial \mathbf{x}^T$ using the chain rule:

$$\left. \frac{\partial \phi}{\partial \mathbf{x}^T} \right|_{\mathbf{x}^*} = \underbrace{\mathbf{I} - \mathbf{J}^{-1} \cdot \underbrace{\frac{\partial \mathbf{F}}{\partial \mathbf{x}^T}}_{\mathbf{J}}}_{\mathbf{0}} - \sum_i \frac{\partial \mathbf{a}_i}{\partial \mathbf{x}^T} \cdot \underbrace{F_i}_{F_i(\mathbf{x}^*)=0} = \mathbf{0} \quad (121)$$

In (121), we have used the definition of \mathbf{F} and the solution \mathbf{F} at \mathbf{x}^* to determine that the derivative $\partial \phi / \partial \mathbf{x}^T$ in the solution point is zero. Inserting (121) in (119), we obtain that the error $\boldsymbol{\varepsilon}$ is of quadratic order. This shows the quadratic convergence of the Newton-Raphson iteration:

$$\boldsymbol{\varepsilon}^{(\mu+1)} = \mathcal{O}\{\boldsymbol{\varepsilon}^{(\mu)} \cdot \boldsymbol{\varepsilon}^{(\mu)T}\} \rightarrow \text{quadratic convergence} \quad (122)$$

The Newton-Raphson iteration is based on a linearization of the function vector \mathbf{F} . Quadratic convergence is only held if the linearization is close enough to the nonlinear function. This is only the case if the start point is sufficiently close to the solution point. As it may not be known if the start point is close enough to the solution, another method may be applied first in the root-finding before switching to Newton-Raphson.

4.5 Termination criteria for the iterative root finding

As we do not know the exact solution values, termination criteria are usually based on the differences between subsequent values of \mathbf{x} and \mathbf{F} , as formulated in (123) and (124), and a maximum number of iteration steps, as developed in (125).

$$\|\mathbf{F}(\mathbf{x}^{(\mu+1)}) - \mathbf{F}(\mathbf{x}^{(\mu)})\| \leq \Delta F \quad (123)$$

$$\|\mathbf{x}^{(\mu+1)} - \mathbf{x}^{(\mu)}\| \leq \Delta x^{(A)} \text{ or } \frac{\|\mathbf{x}^{(\mu+1)} - \mathbf{x}^{(\mu)}\|}{\|\mathbf{x}^{(\mu+1)}\|} \leq \Delta x^{(R)} \quad (124)$$

$$\mu > \mu_{\max} \quad (125)$$

5. The linearized nodal voltage system in iteration step $\mu+1$

Chapter 4 has treated the solution of a nonlinear root-finding problem in general. We will now apply the presented Newton-Raphson iteration method to the specific properties of circuit simulation. The nodal voltage system will be formulated for a Newton-Raphson iteration step. This will be done in two flavors; first, for the complete nonlinear nodal equation system after having composed it. Second, the linearization will be formulated for the individual branches with nonlinear elements before composing the entire nodal voltage system. We will see that the second approach brings two excellent features:

- The Newton-Raphson iteration can be interpreted with an equivalent circuit.
- The linearized equation system can be set up with the rules presented in Sec. 2.4 (and Sec. 2.3 for controlled sources).

5.1 Linearizing the nonlinear nodal system equations

We set up the nodal equations analogous to (23), (25), (27), and (28). The difference is in the branch constitutive equations, which are nonlinear relations between branch currents and branch voltages in the general case:

$$\begin{aligned} \text{(KCL)} \quad & \mathbf{A} \cdot \mathbf{i} = \mathbf{0} (= \mathbf{F}) \\ \text{(BCE)} \quad & \mathbf{i} - \mathbf{I}_0 = \mathbf{g}(\mathbf{u} - \mathbf{U}_0) \text{ with a nonlinear function } \mathbf{g}(\cdot) \\ \text{(KVL)} \quad & \mathbf{u} = \mathbf{A}^T \cdot \mathbf{u}_n \end{aligned} \quad (126)$$

Inserting (27) into (25) and then into (23) according to (28) results in the nonlinear nodal system equations:

$$\mathbf{F}(\mathbf{u}_n) = \mathbf{A} \cdot [\mathbf{g}(\mathbf{A}^T \cdot \mathbf{u}_n - \mathbf{U}_0) + \mathbf{I}_0] = \mathbf{0} \quad (127)$$

Please note that the functions \mathbf{g} and, consequently, \mathbf{F} describe currents.

To prepare an iterative solution of (127) with the Newton-Raphson approach, we linearize (127) around a point $\mathbf{u}_n^{(\mu+1)}$ and set the linearized $\bar{\mathbf{F}}$ to be zero instead of the nonlinear \mathbf{F} :

$$\bar{\mathbf{F}}(\mathbf{u}_n) = \mathbf{F}(\mathbf{u}_n^{(\mu)}) + \mathbf{Y}_n^{(\mu)} \cdot (\mathbf{u}_n - \mathbf{u}_n^{(\mu)}) = \mathbf{0} \quad (128)$$

Here, $\mathbf{Y}_n^{(\mu+1)}$ is the so-called **differential nodal admittance matrix**:

$$\mathbf{Y}_n^{(\mu)} = \left. \frac{\partial \mathbf{F}(\mathbf{u}_n)}{\partial \mathbf{u}_n^T} \right|_{\mathbf{u}_n^{(\mu)}} = \begin{bmatrix} \frac{\partial F_1}{\partial u_{n1}} & \cdots & \frac{\partial F_1}{\partial u_{nn}} \\ \vdots & & \vdots \\ \frac{\partial F_n}{\partial u_{n1}} & \cdots & \frac{\partial F_n}{\partial u_{nn}} \end{bmatrix} \mathbf{u}_n^{(\mu)} \quad (129)$$

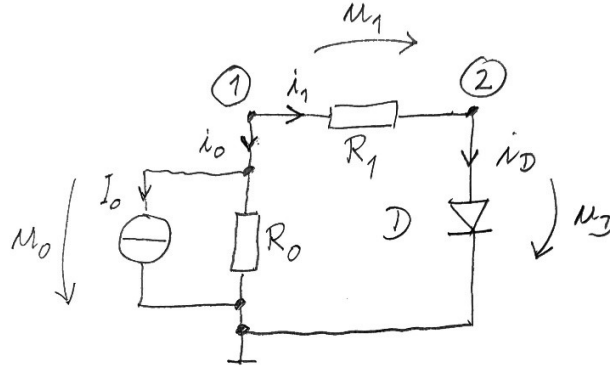
The Newton-Raphson iteration takes the solution of the following linear equation system corresponding to (128) as the next iteration point $\mathbf{u}_n^{(\mu+1)}$:

$$\mathbf{Y}_n^{(\mu)} \cdot \mathbf{u}_n^{(\mu+1)} = \mathbf{Y}_n^{(\mu)} \cdot \mathbf{u}_n^{(\mu)} - \mathbf{F}(\mathbf{u}_n^{(\mu)}) \rightarrow \mathbf{u}_n^{(\mu+1)} \quad (130)$$

Circuit example:

The circuit example in **Figure 35** will be used to illustrate how to set up the linearized nodal voltage system.

Figure 35: Circuit example for setting up the nodal voltage system in iteration step $\mu+1$.



Kirchhoff current law, branch constitutive equations, and Kirchhoff voltage law for this circuit are as follows:

$$\begin{aligned} \text{(KCL)} \quad & i_0 + i_1 = 0 \\ & -i_1 + i_D = 0 \end{aligned} \quad (131)$$

$$\begin{aligned} \text{(BCE)} \quad & i_0 - I_0 = u_0/R_0 \\ & i_1 = u_1/R_1 \\ & i_D = I_s \cdot (e^{u_D/U_T} - 1) \end{aligned} \quad (132)$$

$$\begin{aligned} \text{(KVL)} \quad & u_0 = u_{n1} \\ & u_1 = u_{n1} - u_{n2} \\ & u_D = u_{n2} \end{aligned} \quad (133)$$

Inserting (KVL) into (BCE), marked in green, and (BCE) into (KCL), marked in red, leads to the following nonlinear vector function \mathbf{F} whose root is to be found:

$$\begin{aligned} F_1(\mathbf{u}_n) &= u_{n1}/R_0 + u_{n1}/R_1 - u_{n2}/R_1 + I_0 = 0 \\ F_2(\mathbf{u}_n) &= -u_{n1}/R_1 + u_{n2}/R_1 + I_s \cdot (e^{u_{n2}/U_T} - 1) = 0 \end{aligned} \quad (134)$$

The differential nodal admittance matrix according to (129) is calculated as follows:

$$\mathbf{Y}_n^{(\mu)} = \begin{bmatrix} \frac{\partial F_1}{\partial u_{n1}} & \frac{\partial F_1}{\partial u_{n2}} \\ \frac{\partial F_2}{\partial u_{n1}} & \frac{\partial F_2}{\partial u_{n2}} \end{bmatrix}^{(\mu)} = \begin{bmatrix} \frac{1}{R_0} + \frac{1}{R_1} & -\frac{1}{R_1} \\ -\frac{1}{R_1} & \frac{1}{R_1} + \frac{I_s}{U_T} \cdot e^{\frac{u_{n2}^{(\mu)}}{U_T}} \end{bmatrix} \quad (135)$$

Inserting (135) and (134) into (130), we obtain the following linearized nodal equation system. We have not expanded the differential nodal admittance matrix on the left side of the equation but focused on the right side.

$$\begin{aligned}
 \mathbf{Y}_n^{(\mu)} \cdot \begin{bmatrix} u_{n1} \\ u_{n2} \end{bmatrix}^{(\mu+1)} &= \mathbf{Y}_n^{(\mu)} \cdot \mathbf{u}_n^{(\mu)} - \mathbf{F}(\mathbf{u}_n^{(\mu)}) \\
 &= \begin{bmatrix} \left(\frac{1}{R_0} + \frac{1}{R_1}\right) \cdot u_{n1}^{(\mu)} - \frac{1}{R_1} \cdot u_{n2}^{(\mu)} \\ -\frac{1}{R_1} \cdot u_{n1}^{(\mu)} + \frac{1}{R_1} \cdot u_{n2}^{(\mu)} + \frac{I_S}{U_T} \cdot e^{\frac{u_{n2}^{(\mu)}}{U_T}} \cdot u_{n2}^{(\mu)} \end{bmatrix} \\
 &\quad - \begin{bmatrix} \left(\frac{1}{R_0} + \frac{1}{R_1}\right) \cdot u_{n1}^{(\mu)} - \frac{1}{R_1} \cdot u_{n2}^{(\mu)} + I_0 \\ -\frac{1}{R_1} \cdot u_{n1}^{(\mu)} + \frac{1}{R_1} \cdot u_{n2}^{(\mu)} + I_S \cdot \left(e^{\frac{u_{n2}^{(\mu)}}{U_T}} - 1\right) \end{bmatrix} \\
 &= \begin{bmatrix} -I_0 \\ \frac{I_S}{U_T} \cdot e^{\frac{u_{n2}^{(\mu)}}{U_T}} \cdot u_{n2}^{(\mu)} - I_S \cdot \left(e^{\frac{u_{n2}^{(\mu)}}{U_T}} - 1\right) \end{bmatrix} \quad (136)
 \end{aligned}$$

We observe that many terms of the minuend and subtrahend of the right-hand side eliminate each other. It does not make sense to set them up if they disappear later. Looking at the example, we recognize that mutual elimination happens for the linear elements in the circuit. This suggests to set up the linearization only for the nonlinear elements. Nonlinear elements occur in the branches. Therefore, we will first linearize these elements in the branch constitutive equations (BCE) of (126) and then compose the resulting equations with (KVL) and (KCL). This is described in the following.

5.2 Linearizing the branch constitutive equations

We linearize the branch currents of branches with nonlinear elements:

$$\begin{aligned}
 \mathbf{i}^{(\mu+1)} &= \mathbf{i}^{(\mu)} + \mathbf{Y}^{(\mu)} \cdot (\mathbf{u}^{(\mu+1)} - \mathbf{u}^{(\mu)}) \\
 &= \mathbf{Y}^{(\mu)} \cdot \mathbf{u}^{(\mu+1)} + \mathbf{i}^{(\mu)} - \mathbf{Y}^{(\mu)} \cdot \mathbf{u}^{(\mu)} \quad (137)
 \end{aligned}$$

$$\text{with } \mathbf{i}^{(\mu)} = \mathbf{I}_0 + \mathbf{g}(\mathbf{u}^{(\mu)} - \mathbf{U}_0) \quad (138)$$

\mathbf{i} and \mathbf{u} are the vectors of branch currents and voltages. The matrix \mathbf{Y} is the **differential branch admittance matrix** with a diagonal structure:

$$\mathbf{Y}^{(\mu)} = \frac{\partial \mathbf{i}}{\partial \mathbf{u}^T} \Big|^{(\mu)} = \frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}^T} \Big|^{(\mu)} = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} \Big|^{(\mu)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{\partial g_k}{\partial u_k} \Big|^{(\mu)} \end{bmatrix} \quad (139)$$

Controlled elements will be included analogously to the described way for linear elements in Sec. 2.3.

Please note (138): The currents \mathbf{i} from the previous iteration step μ are calculated with the actual nonlinear function for the respective branches. This corresponds to going back on the nonlinear function after each linearization and updating the linearization, as illustrated in **Figure 33**.

Inserting (137) into (BCE) of (126), we obtain:

$$\underbrace{\frac{\partial \mathbf{F}}{\partial \mathbf{i}^T}}_{\frac{\partial \mathbf{F}}{\partial \mathbf{u}_n^T} \Big|^{(\mu)} = \mathbf{Y}_n^{(\mu)}} \cdot \underbrace{\mathbf{Y}^{(\mu)}}_{\frac{\partial \mathbf{i}}{\partial \mathbf{u}^T} \Big|^{(\mu)}} \cdot \underbrace{\mathbf{A}^T}_{\frac{\partial \mathbf{u}}{\partial \mathbf{u}_n^T}} \cdot \mathbf{u}_n^{(\mu+1)} = \mathbf{A} \cdot \underbrace{[\mathbf{Y}^{(\mu)} \cdot \mathbf{u}_n^{(\mu)} - \mathbf{i}^{(\mu)}]}_{\underbrace{\mathbf{A}^T \cdot \mathbf{u}_n^{(\mu)}}_{\mathbf{Y}_n^{(\mu)} \cdot \mathbf{u}_n^{(\mu)} - \underbrace{\mathbf{A} \cdot \mathbf{i}^{(\mu)}}_{\mathbf{F}(\mathbf{u}_n^{(\mu)})}}]} \quad (140)$$

The braces underneath the respective terms in (140) use earlier equations to transform (140) such that it becomes (130) to confirm that they are identical. While the result is identical, they differ in the point in time when the linearization takes place. For (130), the system equations were set up first, and then the linearization was done. For (140), first, the linearization was done (where necessary), and then the system equations were composed.

Circuit example:

We will illustrate the proceeding of linearization in branches before setting up the system equations with the example of **Figure 35**. We have to linearize the diode current:

$$i_D^{(\mu+1)} = i_D^{(\mu)} + \underbrace{\frac{\partial i_D}{\partial u_D} \Big|^{(\mu)}}_{Y_D^{(\mu)} = \frac{I_S}{U_T} \cdot e^{\frac{u_D^{(\mu)}}{U_T}}} \cdot (u_D^{(\mu+1)} - u_D^{(\mu)}) \quad (141)$$

$$= Y_D^{(\mu)} \cdot u_D^{(\mu+1)} + i_D^{(\mu)} - Y_D^{(\mu)} \cdot u_D^{(\mu)} \quad (142)$$

Please note how (142) resembles (137). Eq. (142) is now inserted into (134) instead of the nonlinear current function i_D . This leads to:

$$\begin{aligned} u_{n1}/R_0 + u_{n1}/R_1 - u_{n2}/R_1 + I_0 &= 0 \\ -u_{n1}/R_1 + u_{n2}/R_1 + Y_D^{(\mu)} \cdot u_{n2}^{(\mu+1)} + i_D^{(\mu)} - Y_D^{(\mu)} \cdot u_{n2}^{(\mu)} &= 0 \end{aligned} \quad (143)$$

Please note that (143) represents the linear equation system to compute the node voltage in step $\mu+1$. Hence, the node voltages without iteration index in (143) obtain a superscript $(\mu+1)$. Using the definition of $\mathbf{Y}_n^{(\mu)}$ in (135) and $\mathbf{Y}_D^{(\mu)}$ in (141), we can formulate (143) in the following way:

$$\mathbf{Y}_n^{(\mu)} \cdot \mathbf{u}_n^{(\mu)} = \begin{bmatrix} -I_0 \\ Y_D^{(\mu)} \cdot u_{n2}^{(\mu)} - i_D^{(\mu)} \end{bmatrix} \quad (144)$$

Comparing the second entries on the right sides of (136) and (144), we can confirm that they are identical. That means (136) and (144) are identical. But (144) has been obtained straightforwardly without setting up terms only to eliminate them later.

5.3 Equivalent circuit model for linearized branch constitutive equations

We will now formulate the linearization for a Newton-Raphson iteration of a nonlinear branch element with a generic nonlinear u-i relation:

$$i - I_0 = g(u - U_0) \quad (145)$$

The Newton-Raphson iteration yields a generic iteration function in iteration step $\mu+1$:

$$i^{(\mu+1)} = i^{(\mu)} + \underbrace{\frac{\partial i}{\partial u}}_{Y^{(\mu)}} \Big|^{(\mu)} \cdot (u^{(\mu+1)} - u^{(\mu)}) \quad (146)$$

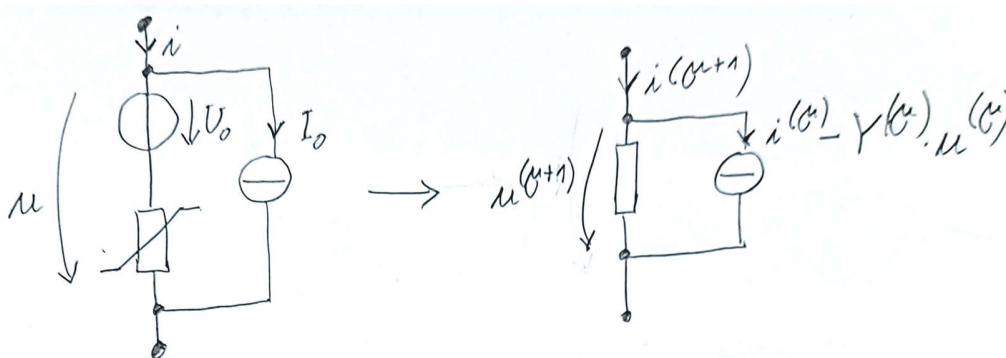
$$= Y^{(\mu)} \cdot u^{(\mu+1)} + i^{(\mu)} - Y^{(\mu)} \cdot u^{(\mu)} \quad (147)$$

$$\text{with } i^{(\mu)} = I_0 + g(u^{(\mu)} - U_0) \quad (148)$$

These equations generalize the diode example's equations (141) and (142).

Equation (147) can be interpreted with a generic equivalent circuit composed of a parallel combination of an admittance and a current source, as shown in **Figure 36**.

Figure 36: Equivalent circuit model for nonlinear branch element in iteration step $\mu+1$.



The equivalent circuit model of the Newton-Raphson iteration enables excellent features for DC simulation:

- The set-up of the (modified) nodal system works in the same way as defined in Sec. 2.4.
- The solution methods established for linear equation systems can be applied for one iteration step of the Newton-Raphson iteration.
- The equivalent circuit model provides a technical interpretation of the mathematical root-finding problem in circuit simulation and a unified circuit-based view.
- Further transformations may enable problem simplification.

Direct set-up of nodal voltage system for example 1

We will use the equivalent circuit model to set up the nodal voltage system for the example in **Figure 35**.

From (141), (142), and (148), we have for the diode the following equations:

$$i_D^{(\mu+1)} = Y_D^{(\mu)} \cdot u_D^{(\mu+1)} + I_D^{(\mu)} \quad (149)$$

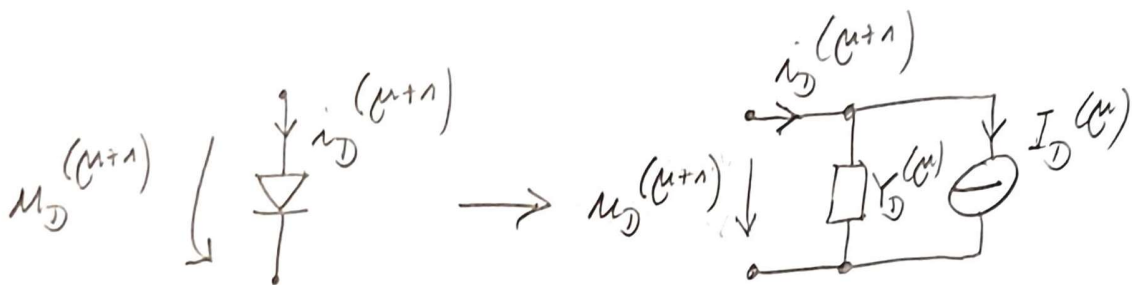
$$Y_D^{(\mu)} = \frac{I_S}{U_T} \cdot e^{\frac{u_D^{(\mu)}}{U_T}} \quad (150)$$

$$I_D^{(\mu)} = i_D^{(\mu)} - Y_D^{(\mu)} \cdot u_D^{(\mu)} \quad (151)$$

$$i_D^{(\mu)} = I_S \cdot (e^{\frac{u_D^{(\mu)}}{U_T}} - 1) \quad (152)$$

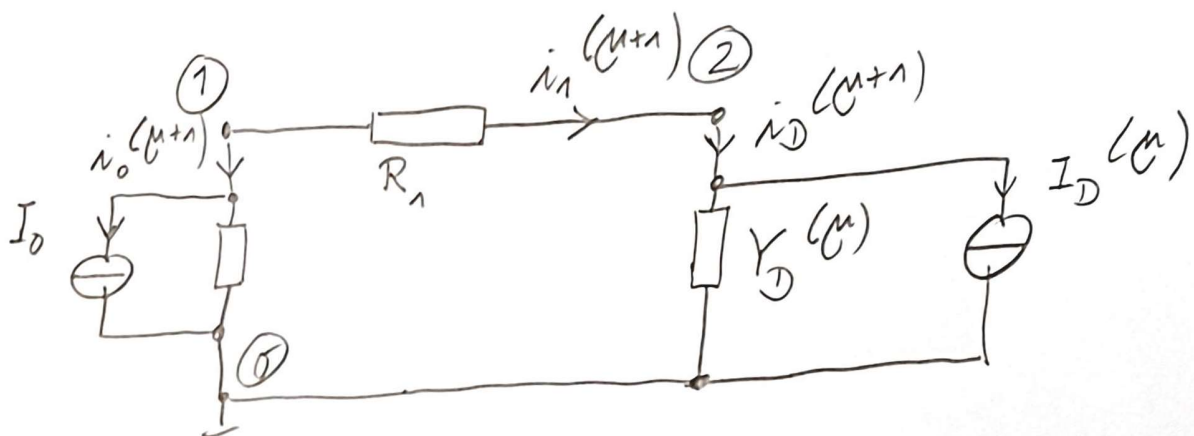
Eq. (151) introduces another abbreviation. This yields the following equivalent circuit in **Figure 37**.

Figure 37: Equivalent circuit for the diode in Figure 35.



Inserting **Figure 37** in **Figure 35** leads to an overall equivalent circuit in **Figure 38**.

Figure 38: Equivalent circuit in iteration step $\mu+1$ of the circuit in Figure 35.



Please note that the Newton-Raphson iteration computes new values in step $\mu+1$ from values from the previous iteration step μ . The values from step μ are represented by constant elements in the equivalent circuit. The values of currents and voltages to be determined in the current iteration step are then marked with $\mu+1$.

The rules of Sec. 2.4 are applied to the equivalent circuit in **Figure 38**. Here, we use that $u_D = u_{n2}$. This leads to the following linear equation system for iteration step $\mu+1$:

$$\begin{bmatrix} \frac{1}{R_0} + \frac{1}{R_1} & -\frac{1}{R_1} \\ -\frac{1}{R_1} & \frac{1}{R_1} + Y_D^{(\mu)} \end{bmatrix} \cdot \begin{bmatrix} u_{n1} \\ u_{n2} \end{bmatrix}^{(\mu+1)} = \begin{bmatrix} -I_0 \\ -I_D^{(\mu)} \end{bmatrix} \quad (153)$$

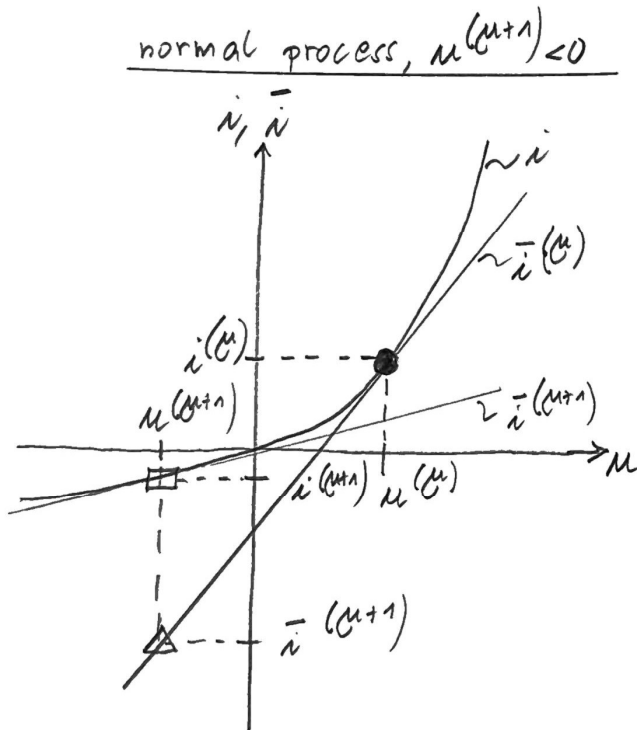
Applying the corresponding abbreviations, we can confirm that (136), (143), (144) and (153) are identical.

Setting up (153) through equivalent circuit models for nonlinear circuit elements in branches represents the straightforward way for both AC and DC simulation. We will discover that we can establish an analogous way of setting up the nodal voltage system for TR simulation by interpreting the required mathematical numerical integration with corresponding equivalent circuit models.

5.4 Modified Newton-Raphson method

The Newton-Raphson method needs to start “close enough” to the solution not only to converge at a quadratic convergence rate but to converge at all. The stronger the nonlinearity of a function, the shorter the required step length of a Newton iteration to maintain convergence. One approach to cope with the degree of nonlinearity is step length control. Besides that, there are functions with known regions of stronger or weaker nonlinearity, like the exponential function. In an area of stronger nonlinearity, the inverse function can represent a benign alternative for linearization. This path is called the modified process. The two options are illustrated for an exponential function in **Figure 39**. On the left, the regular process is shown. The current i is given as an exponential function of voltage u . The linearization $\bar{i}^{(\mu)}$ at the current point $i^{(\mu)}, u^{(\mu)}$, marked by the black dot, is the basis for the linearized equation system of the complete circuit. The solution of the complete equation system in this iteration step may lead to the point $\bar{i}^{(\mu+1)}, u^{(\mu+1)}$ on the linearization $\bar{i}^{(\mu)}$ marked with a triangle. For the next iteration step, we go back to the nonlinear function. As we use the exponential function, we determine the true current $i^{(\mu+1)}$ at $u^{(\mu+1)}$, this leads to the point marked with a box. Then, the process continues with a new linearization. Imagine the triangle point would have been on the right of the voltage $u^{(\mu)}$. Going vertically up to get back on the exponential function in this region would have led to a current very different from the linear model. Convergence would have been jeopardized. Going horizontally to the nonlinear exponential function would be much better in this case. This corresponds to building the inverse function and applying the modified process illustrated on the right of **Figure 39**. The inverse function gives the voltage u as a function of current i with a logarithmic function. With the inverse function we get from the triangle point to the box point by moving from $(i^{(\mu+1)}, \bar{u}^{(\mu+1)})$ to $(i^{(\mu+1)}, u^{(\mu+1)})$ (instead of moving from $(\bar{i}^{(\mu+1)}, u^{(\mu+1)})$ to $(i^{(\mu+1)}, u^{(\mu+1)})$ in the normal process).

Figure 39: Modified Newton-Raphson method.

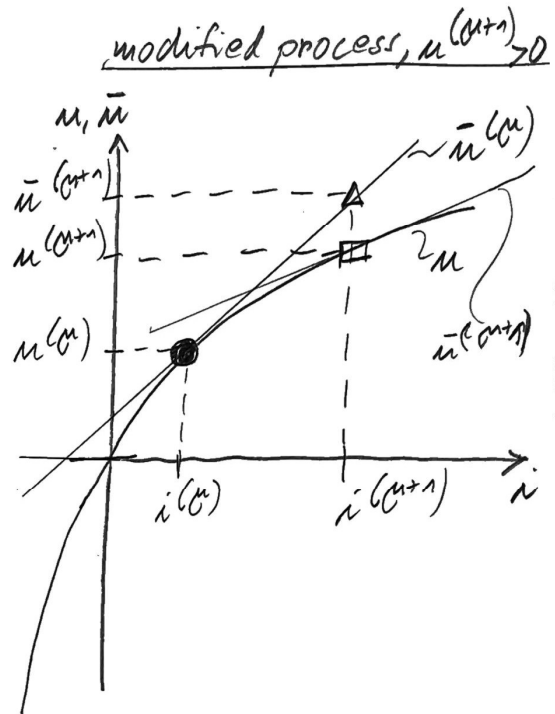


● → △ → □

$$i = I_s \cdot (e^{\mu/V_T} - 1)$$

$$\bar{i}^{(u)} = i^{(u)} + \left. \frac{\partial i}{\partial \mu} \right|^{(u)} \cdot (\mu - \mu^{(u)})$$

$$\gamma^{(u)} = \frac{I_s}{V_T} \cdot e^{\mu^{(u)}/V_T}$$



● → △ → □

$$\mu = V_T \cdot \ln\left(1 + \frac{i}{I_s}\right)$$

$$\bar{\mu}^{(u)} = \mu^{(u)} + \left. \frac{\partial \mu}{\partial i} \right|^{(u)} \cdot (i - i^{(u)})$$

$$Z^{(u)} = \frac{V_T}{i^{(u)} + I_s}$$

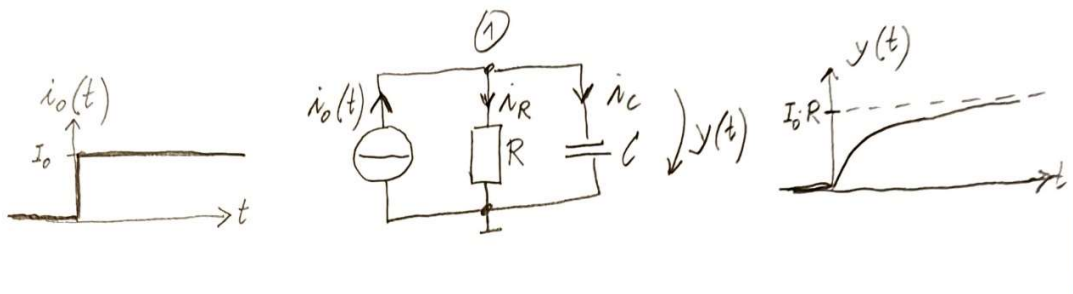
6. TR analysis

Transient analysis (in German also: *Einschwinganalyse*) aims to analyze the transient response of a circuit to a transient excitation at the input(s). This can refer to the transition to the DC operating point after switching on the power or the simulation of transient performance features like slew rate or delay. The circuit contains nonlinear devices like transistors and energy-storage devices like capacitors and inductors. Transient simulation of nonlinear circuits requires the numerical integration of nonlinear differential-algebraic equations. In the following, we will recapitulate continuous and discrete solutions for linear circuits, which can be calculated analytically. When it comes to nonlinear circuits, a numerical solution is necessary. We will give an example of how the stability and accuracy of numerical integration come into play. After that, we will present four simple numerical integration methods. These are Forward Euler (*expliziter Euler*), Backward Euler (*impliziter Euler*), Trapezoidal method (*Trapez-Methode*), and Gear method of 2nd order (*Gear-Methode zweiter Ordnung*). Fundamental properties of these methods will be discussed.

6.1 Determining the continuous step response of a simple linear circuit

The example in **Figure 40** will be used. It is a parallel composition of a resistor and a capacitor, shown in the middle. The input current $i_0(t)$ steps to the value I_0 at time zero, as shown on the left. We want to compute the continuous step response $y(t)$ at the capacitor, shown on the right.

Figure 40: Continuous step response of an example circuit.



Kirchhoff current law at node 1, together with the simple ordinary differential equation for the capacitor, gives the following overall ordinary differential equation (ODE) for the circuit.

$$\begin{aligned} i_c + i_R &= i_0(t) \\ \Leftrightarrow C \cdot \dot{y}(t) + \frac{1}{R} \cdot y(t) &= i_0(t) \\ \Leftrightarrow \dot{y}(t) + \frac{1}{R \cdot C} \cdot y(t) &= \frac{R}{R \cdot C} \cdot i_0(t) \end{aligned} \quad (154)$$

An analytical and explicit solution of the ODE (154) is possible because of the linearity of the devices. A Laplace transformation is applied to (154). This results in the following Laplace transform.

$$(154) \circ \bullet p \cdot Y(p) - y(+0) + \frac{1}{R \cdot C} \cdot Y(p) = \frac{R}{R \cdot C} \cdot \frac{I_0}{p} \quad (155)$$

We assume that no energy is stored and abbreviate the pole p_∞ , i.e.,

$$y(+0) = 0, \quad p_\infty = \frac{-1}{R \cdot C} \quad (156)$$

to obtain the Laplace transform Y of the voltage y in explicit form:

$$Y(p) = \frac{-p_\infty \cdot R \cdot I_0}{p \cdot (p - p_\infty)} \quad (157)$$

Eq. (157) is decomposed into partial fractions according to the poles 0 and p_∞ in the denominator:

$$Y(p) = \frac{A}{p} + \frac{B}{p - p_\infty} \quad (158)$$

$$A = \lim_{p \rightarrow 0} p \cdot Y(p) = R \cdot I_0 \quad (159)$$

$$B = \lim_{p \rightarrow p_\infty} (p - p_\infty) \cdot Y(p) = -R \cdot I_0 \quad (160)$$

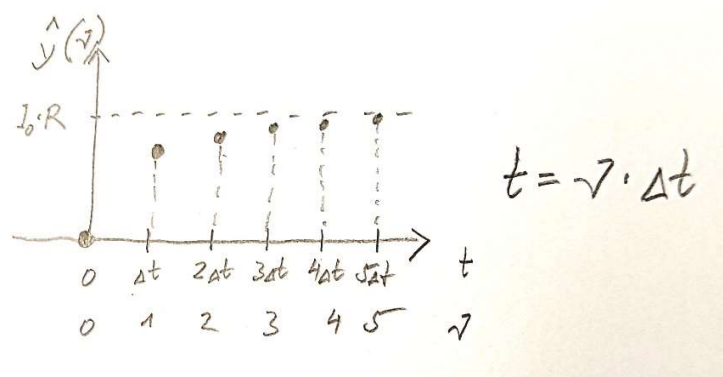
For each partial fraction, a Laplace back transform exists in a library so that the overall Laplace back transform of the voltage y can be composed:

$$Y(p) \bullet \longrightarrow y(t) = R \cdot I_0 \cdot (1 - e^{p_\infty \cdot t}) \quad (161)$$

6.2 Determining the discrete step response of a simple linear circuit

The same example in **Figure 40** will be used. First, the continuous time t is discretized into discrete time steps ν of the same distance Δt . The response will only be computed at these discrete time steps, as illustrated in **Figure 41**.

Figure 41: Discrete step response of the example circuit in Figure 40.



The ODE (154) then becomes a function of discrete time steps ν using the abbreviation p_∞ in (156):

$$\hat{y}(\nu) = p_\infty \cdot y(\nu) - p_\infty \cdot R \cdot i_0(\nu) \quad (162)$$

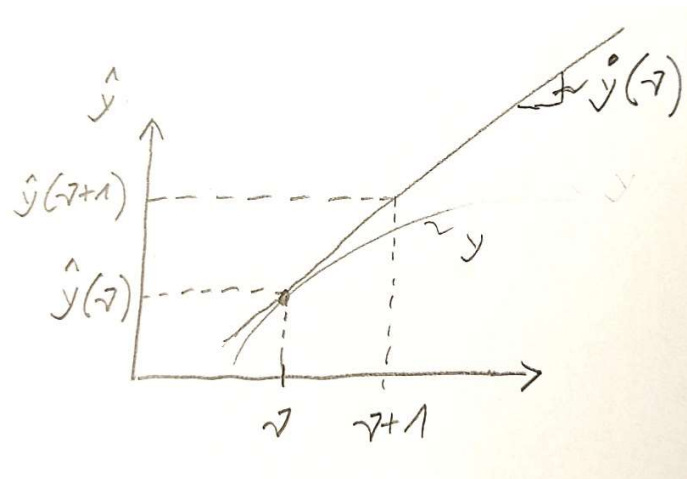
Integrating a discretized space needs approximation because the actual curvature between two steps is ignored. The integration steps are based on grid points of the function y and the function \dot{y} from the actual time step and past time steps. These grid points are combined in particular difference equations.

One such difference equation is the so-called Forward Euler. It reads as follows:

$$\hat{y}(\nu + 1) = \hat{y}(\nu) + \Delta t \cdot \dot{y}(\nu) \quad (163)$$

We have introduced the hat over y to denote the approximation that comes with the numerical integration. Eq. (163) can be interpreted as a forward approximation of y at time $\nu+1$ by the derivative \dot{y} at time ν . This is illustrated in **Figure 42**.

Figure 42: Visual interpretation of Forward Euler difference equation.



The term $\dot{y}(\nu)$ in the Forward Euler difference equation (163) is now replaced by the discretized ODE (162). This results in a difference equation of the complete circuit:

$$\hat{y}(\nu + 1) = (1 + p_\infty \cdot \Delta t) \cdot \hat{y}(\nu) - p_\infty \cdot \Delta t \cdot R \cdot i_0(\nu) \quad (164)$$

Eq. (164) formulates a successive computation of the voltage y starting from the (given) initial value $y(0)$:

$$\hat{y}(\nu), \quad \nu = 1, 2, 3, \dots \quad (165)$$

In this case of a linear system, we can determine an explicit and analytical formula, which gives the value of y at any time point ν without the need for successive computation. This is achieved with the help of a z -transformation of the circuit's difference equation (164):

$$(164) \circ \bullet z \cdot \hat{Y}(z) - z \cdot y(0) = (1 + p_\infty \cdot \Delta t) \cdot \hat{Y}(z) - p_\infty \cdot \Delta t \cdot R \cdot \frac{z \cdot I_0}{z - 1} \quad (166)$$

Again, we assume no energy is stored, i.e., $y(0)=0$. We solve (166) for \hat{Y} and perform a partial fraction decomposition:

$$\hat{Y}(z) = \frac{-p_\infty \cdot \Delta t \cdot R}{z - (1 + p_\infty \cdot \Delta t)} \cdot \frac{z \cdot I_0}{z - 1} = \frac{z \cdot R \cdot I_0}{z - 1} - \frac{z \cdot R \cdot I_0}{z - (1 + p_\infty \cdot \Delta t)} \quad (167)$$

The back transformation of the two partial fractions in (167) finally leads to the explicit discrete approximation of the voltage $\hat{y}(\nu)$ based on numerical integration with the Forward Euler approach:

$$\hat{Y}(z) \bullet \circ \hat{y}(\nu) = [1 - (1 + p_\infty \cdot \Delta t)^\nu] \cdot R \cdot I_0 \quad (168)$$

6.3 First glance at error and stability of numerical integration

We will look at the example circuit in **Figure 40** and compare the numerical integration with the Forward Euler approach, represented by (168), with the exact continuous integration, represented by (161). To make it easier, we define $I_0 \cdot R=1$ and $p_\infty = -1$. This simplifies (161) and (168) to:

$$\text{exact: } y(t) = 1 - e^{-t} \quad (169)$$

$$\text{Forward Euler: } \hat{y}(\nu) = [1 - (1 + p_\infty \cdot \Delta t)^\nu] = [1 - (1 - \Delta t)^\nu] \quad (170)$$

Let us look at the point in time $t=1$. The exact value of the voltage at the capacitor is, according to (169):

$$y(1) = 1 - e^{-1} = 0.632121 \quad (171)$$

The numerical value by Forward Euler numerical integration is, according to (170):

$$\hat{y}(1) = 1 - (1 - \Delta t)^\nu, \quad \nu \cdot \Delta t = 1 \quad (172)$$

We observe that the numerical value depends on the choice of Δt . The following table shows the resulting values of $\hat{y}(1)$ for three different values of Δt and the difference (error) $\varepsilon^{(A)}$ between $\hat{y}(1)$ and $y(1)$.

Δt	ν	$\hat{y}(1)$	$\varepsilon^{(A)} = \hat{y}(1) - y(1) $
0.1	10	0.651322	0.019201
0.05	20	0.641514	0.009393
0.025	40	0.636768	0.004647

The error reduces proportionally to the time step, $\varepsilon^{(A)} \sim \Delta t$, i.e., the smaller the time step, the smaller the error. It seems that half the time step results in half the error. The dependence of the error on the time step results from the mentioned approximation of the integration by ignoring the actual function between two time steps. It is intuitively clear that this error somehow increases if the range Δt that is not "illuminated" increases.

There is another issue with numerical integration. We know from (169) that the voltage converges to $y=1$ for $t \rightarrow \infty$. Eq. (170) only mirrors this behavior if $|1 - \Delta t| < 1$. We call this stability. Therefore, in this example:

$$\begin{aligned} \text{algorithm stable: } & 0 < \Delta t < 2 \\ \text{algorithm unstable: } & \Delta t \geq 2 \end{aligned} \quad (173)$$

6.4 Mathematical task of TR analysis

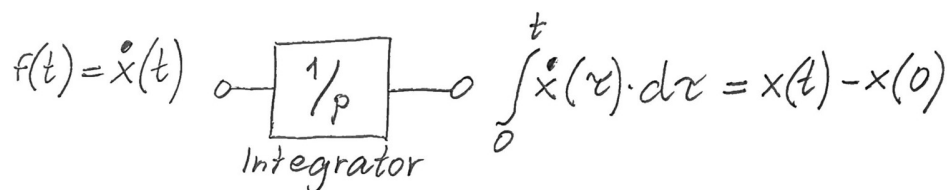
The mathematical task of transient analysis is the solution of linear or nonlinear state equations of an electronic system. These state equations are:

<u>linear state equations</u>	<u>nonlinear state equations</u>	
$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{b} \cdot u(t)$	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$	(174)
$y(t) = \mathbf{c}^T \cdot \mathbf{x}(t) + d \cdot u(t)$	$y(t) = g(\mathbf{x}(t), t)$	

The input u is a single input in (174); it may also become a multi-input. The input u has been included in the states x for the nonlinear system description. The single output y may also become a multi-output. The nonlinear state equations are given as nonlinear ordinary differential equations in normal form. The output function is a nonlinear algebraic equation.

The mathematical task of TR analysis is numerical integration, which can be given in a structural description of an integrator in **Figure 43**. The $1/p$ in the integrator system reflects the integration's counterpart after the Laplace transformation.

Figure 43: Integrator.

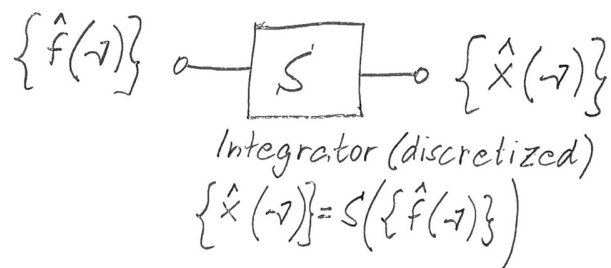


In the case of discretizing the time, the state equations of (174) take the following form:

<u>discretized linear state equations</u>	<u>discretized nonlinear state equations</u>	
$\hat{\mathbf{f}}(\nu) = \mathbf{A} \cdot \hat{\mathbf{x}}(\nu) + \mathbf{b} \cdot u(\nu)$	$\hat{\mathbf{f}}(\nu) = \mathbf{f}(\hat{\mathbf{x}}(\nu), \nu \cdot \Delta t)$	(175)
$\hat{y}(\nu) = \mathbf{c}^T \cdot \hat{\mathbf{x}}(\nu) + d \cdot u(\nu)$	$\hat{y}(\nu) = g(\hat{\mathbf{x}}(\nu), \nu \cdot \Delta t)$	

The structural description of the discretized integrator is given in **Figure 44**.

Figure 44: Integrator (discretized).



6.5 Four properties of numerical integration methods

We will analyze four different properties of four numerical integration methods. These four properties are:

Explicit/implicit: In an implicit method, f and x of the current time step are used. This increases the computational effort for an integration step.

Number of steps: How many previous time steps are used in the difference equation of the numerical integration method?

Accuracy for $|p_\infty \cdot \Delta t| \ll 1$: What is the order k of the error $\epsilon^{(A)} = |\hat{x}(v) - x(v)| \sim \Delta t^k$

Stability: Is the algorithm stable for stable systems, i.e., $\text{Re}\{p_\infty\} < 0$?
Is the algorithm asymptotically stable for $\Delta t \rightarrow \infty$?

6.6 The test differential equation

We will analyze the mentioned properties for a particular differential equation:

$$\dot{x}(t) = p_\infty \cdot x(t), \quad x(0) \neq 0 \quad (176)$$

Its solution is

$$x(t) = x(0) \cdot e^{p_\infty \cdot t} \quad x(\nu) = x(0) \cdot e^{p_\infty \cdot \nu \cdot \Delta t} \quad (177)$$

and describes a decaying sinusoidal oscillation of a stable system with negative real part α_∞ of the system eigenvalue/pole p_∞ :

$$p_\infty = \text{Re}\{p_\infty\} + j \cdot \text{Im}\{p_\infty\} = \alpha_\infty + j \cdot \omega_\infty \quad (178)$$

Electronic systems feature switching between stable states with a transition with or without oscillation. Several poles and zeroes determine the behavior in a system transfer function. Therefore, the test differential equation (176) models the fundamental behavior of electronic circuits and has a role model to analyze numerical integration properties in circuit simulation.

6.7 The difference equations of four simple numerical integration methods

The following will state the difference equations for four simple numerical integration methods. For simplicity, we will concentrate on the one-dimensional case. This is ok, as we will set up the numerical integration for each edge individually in the circuit. We will interpret the formulas visually. To remind us, we deal with the solution of nonlinear, implicit differential equations of the form

$$\dot{x}(t) = f(x(t), t) \text{ with } x(0) = 0 \quad (179)$$

We are aiming to solve the integral

$$x(t) = \int_0^t \dot{x}(\tau) d\tau \quad (180)$$

This is done from time step to time step:

$$x(t + \Delta t) = \int_0^t \dot{x}(\tau) d\tau + \int_t^{t+\Delta t} \dot{x}(\tau) d\tau = x(t) + \int_t^{t+\Delta t} \dot{x}(\tau) d\tau \quad (181)$$

The core task of a numerical integration method is the formula to determine this incremental integration step. It can basically be described as multiplying the width of the time step, Δt , with an approximate finite differential $\widetilde{\Delta x}/\Delta t$:

$$x(\nu + 1) \approx x(\nu) + \widetilde{\Delta x} = x(\nu) + \Delta t \cdot \frac{\widetilde{\Delta x}}{\Delta t} \quad (182)$$

The goal is to model this finite differential to target the correct incremental integration value.

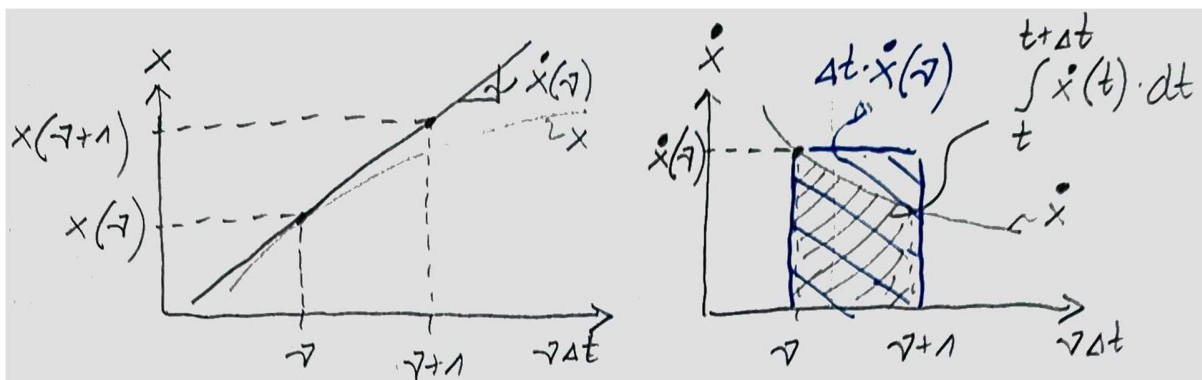
6.7.1 Forward Euler method's difference equation

The Forward Euler (*Expliziter Euler*) method applies the value at time point ν of the function f to be integrated, i.e., the gradient of the target variable x , as approximate finite differential:

$$x(\nu + 1) \approx x(\nu) + \Delta t \cdot \dot{x}(\nu) \quad (183)$$

This is illustrated in **Figure 45**. On the left, the target variable x , i.e., the output of the function to be integrated, is given over time. On the right, the function f to be integrated, i.e., \dot{x} , is shown over time. On the left, the gradient of x with regard to time t at discrete time point ν is used to determine the value of x at time point $\nu+1$.

Figure 45: Visual interpretation of the Forward Euler method.



On the right of **Figure 45**, the value of \dot{x} at time point ν is shown as a function for the integration from ν to $\nu+1$. The blue hatched area is the integrated value that is obtained. It overestimates the true value indicated by the black hatched area in this example.

The **Forward Euler method is one-step** because values from one past time step are included in the difference equation.

6.7.2 Backward Euler method's difference equation

The Backward Euler (*Impliziter Euler*) method applies the value at the next time point $v+1$ of the function f to be integrated, i.e., the gradient of the target variable x "ahead" as approximate finite differential:

$$x(\nu + 1) \approx x(\nu) + \Delta t \cdot \dot{x}(\nu + 1) \quad (184)$$

It is an implicit approach that projects from the future (next discrete time step) backward to the current time step. We will see that an implicit approach increases the computational effort, as implicit linear or nonlinear equations must be solved.

Figure 46: Visual interpretation of the Backward Euler method.

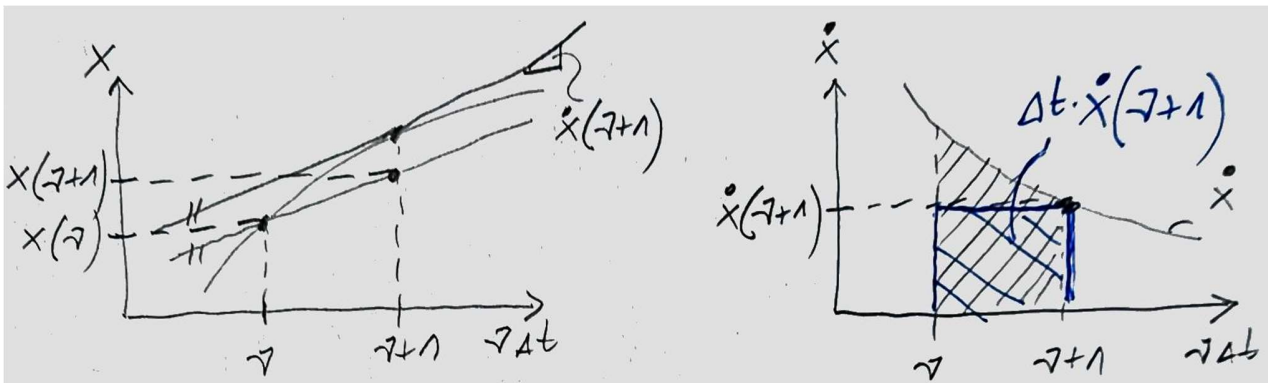


Figure 46 illustrates (184). On the left, in the "space" of x , we see that the gradient \dot{x} is taken in the next time point $v+1$ to project to $x(v+1)$ from time point v with this future gradient. On the right, in the "space" of \dot{x} , we take the future value $\dot{x}(v+1)$ to integrate from v to $v+1$. The resulting blue hatched area underestimates the true integration area hatched in black in this example.

The **Backward Euler method is one-step** because values from one past time step are included in the difference equation.

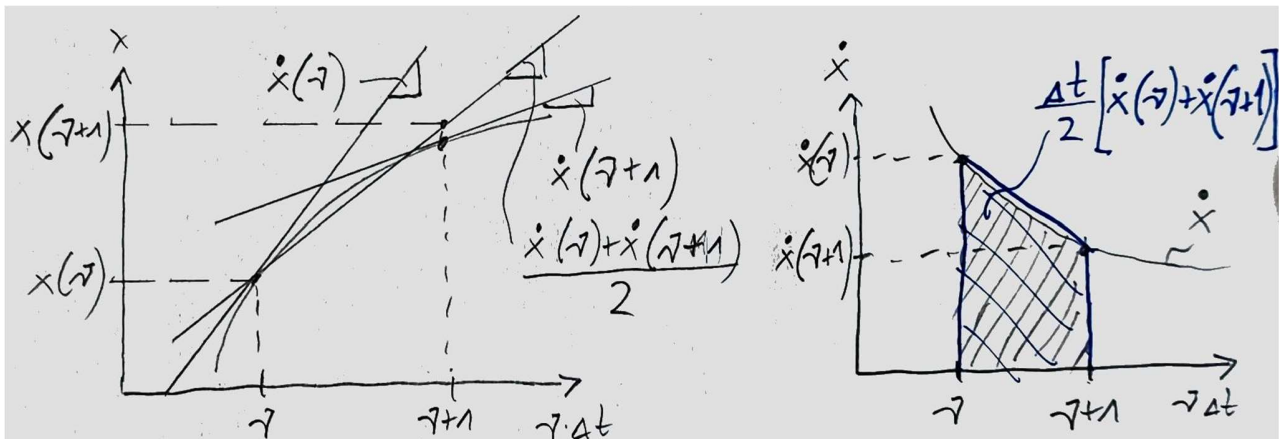
6.7.3 Trapezoidal method's difference equation

As Forward Euler and Backward Euler result in some overestimation or underestimation, an apparent next try for a numerical integration method is to average between the values of $\dot{x}(v)$ and $\dot{x}(v+1)$. This is called the Trapezoidal method:

$$x(\nu + 1) \approx x(\nu) + \Delta t \cdot \frac{1}{2} [\dot{x}(\nu) + \dot{x}(\nu + 1)] \quad (185)$$

Figure 47 illustrates the Trapezoidal method. On the left, the average of the two gradients gives a linear model that gets closer to the real value $x(v+1)$. On the right, the averaging of the two gradients corresponds to the blue-hatched trapezoid, hence the method's name. The true incremental integration step is still overestimated in the example, but the numerical step is much closer to the true value than before.

Figure 47: Visual interpretation of the Trapezoidal method.



The **Trapezoidal method is one-step** because values from one past time step are included in the difference equation.

6.7.4 Second-order Gear method's difference equation

There are more complicated numerical integration formulas. They take more time steps from the past into account and combine implicit values of f/\dot{x} of the next time step with values of \dot{x} and finite differences of x values from the past. One example is the second-order Gear method:

$$x(\nu + 2) \approx x(\nu + 1) + \Delta t \cdot \frac{1}{3} \left[\underbrace{\frac{x(\nu + 1) - x(\nu)}{\Delta t}}_{\text{secant over previous 2 steps}} + 2 \cdot \underbrace{\dot{x}(\nu + 2)}_{\text{derivative in current step}} \right] \quad (186)$$

$$= \frac{4}{3}x(\nu + 1) - \frac{1}{3}x(\nu) + \frac{2}{3}\Delta t \cdot \dot{x}(\nu + 2) \quad (187)$$

It uses values from two past steps ν , $\nu+1$ to compute the current time step $\nu+2$. From the current time step, the derivative \dot{x} is used. The computed integral values x from the recent two steps approximate a derivative by finite difference approach. The derivative $\dot{x}(\nu+2)$ is weighted twice compared to the secant part.

The **second-order Gear method is two-step** because values from two past time steps are included in the difference equation.

6.8 Properties of the four simple numerical integration methods

6.8.1 Forward Euler method's properties

Structure

A structural interpretation or implementation of the Forward Euler method is obtained by a z-transform of its difference equation (183):

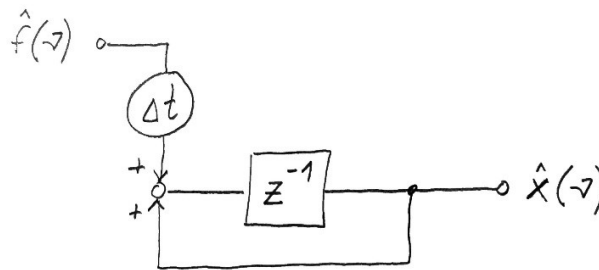
$$(183) \wedge x(0) = 0 \circ \bullet z \cdot \hat{X}(z) = \hat{X}(z) + \Delta t \cdot \hat{F}(z) \quad (188)$$

The z-transform (188) can be reformulated in two ways:

$$(188) \Leftrightarrow \hat{X}(z) = \frac{\Delta t}{z-1} \cdot \hat{F}(z) = z^{-1} \cdot (\hat{X}(z) + \Delta t \cdot \hat{F}(z)) \quad (189)$$

The first expression describes the linear transfer function of a discrete system for numerical integration. The second expression allows us to verify the structure of the Forward Euler numerical integration method in **Figure 48**.

Figure 48: Structure of the Forward Euler numerical integration method.



It represents a hardware/system structure/implementation/interpretation of the Forward Euler method.

State equations

Inserting the discrete state equations given in (175) for \dot{x} in the multivariate version of the difference equations (183) of the Forward Euler method results in the following numerical integration formulas for the Forward Euler method. For linear state equations, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \Delta t \cdot [\mathbf{A} \cdot \hat{\mathbf{x}}(\nu) + \mathbf{b} \cdot u(\nu)] \quad (190)$$

$$= [\mathbf{I} + \Delta t \cdot \mathbf{A}] \cdot \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{b} \cdot u(\nu) \quad (191)$$

For the nonlinear case, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}(\nu), \Delta t) \quad (192)$$

The **Forward Euler is an explicit method**; the next integration value results explicitly from inserting values on the right side of the difference equation.

Eqs. (191) and (192) clarify the explicit character of the Forward Euler method. Inserting available values from the past time step ν on the right side immediately results in the value to be computed in time step $\nu+1$.

Test differential equation

As mentioned, we will investigate several properties of numerical integration methods based on the test differential equation, a fundamental and characteristic differential equation of electronic circuits.

We insert the test differential equation (176) into the difference equation of the Forward Euler method (183) and obtain the following:

$$\hat{x}(\nu + 1) = \hat{x}(\nu) + \Delta t \cdot p_{\infty} \cdot \hat{x}(\nu) = (1 + p_{\infty} \cdot \Delta t) \cdot \hat{x}(\nu) \quad (193)$$

Eq. (193) can also be written as follows:

$$\hat{x}(\nu) = (1 + p_{\infty} \cdot \Delta t)^{\nu} \cdot x(0) \quad (194)$$

Accuracy

The error that the Forward Euler method produces on the test differential equation is obtained as the difference between the numerical solution (194) and the exact solution (177):

$$\varepsilon/x(0) = (1 + p_{\infty} \cdot \Delta t)^{\nu} - e^{p_{\infty} \cdot \Delta t \cdot \nu} = e^{\nu \cdot \ln(1 + p_{\infty} \cdot \Delta t)} - e^{p_{\infty} \cdot \Delta t \cdot \nu} \quad (195)$$

For $|p_{\infty} \cdot \Delta t|$ sufficiently smaller than 1, we can write the following:

$$\ln(1 + p_{\infty} \cdot \Delta t) = p_{\infty} \cdot \Delta t - \frac{(p_{\infty} \cdot \Delta t)^2}{2} + \dots \approx p_{\infty} \cdot \Delta t \cdot \left(1 - \frac{p_{\infty} \cdot \Delta t}{2}\right) \quad (196)$$

We insert (196) into (195) and obtain the following:

$$\varepsilon/x(0) = \varepsilon' \approx e^{\overbrace{p_{\infty} \cdot \nu \cdot \Delta t}^T \cdot \left(1 - \overbrace{\frac{p_{\infty} \cdot \Delta t}{2}}^{\delta}\right)} - e^{p_{\infty} \cdot \nu \cdot \Delta t} = e^{T(1+\delta)} - e^T \quad (197)$$

The abbreviations T , δ , and ε' have been introduced to reduce the writing. We develop (197) into a Taylor series up to the first-order term to obtain the following:

$$\begin{aligned} \varepsilon' &\approx \varepsilon'|_{\delta=0} + \left. \frac{\partial \varepsilon'}{\partial \delta} \right|_{\delta=0} \cdot \delta + \dots \approx 0 + T \cdot e^{T(1+\delta)} \Big|_{\delta=0} \cdot \delta = T \cdot e^T \cdot \delta \\ &\approx p_{\infty} \cdot \nu \cdot \Delta t \cdot e^{p_{\infty} \cdot \nu \cdot \Delta t} \cdot \left(-\frac{p_{\infty} \cdot \Delta t}{2}\right) \end{aligned} \quad (198)$$

Eq. (198) can be rearranged in the following form:

$$\varepsilon \approx -\Delta t \cdot \frac{\overbrace{\nu \cdot \Delta t}^t}{2} \cdot x(0) \cdot p_\infty^2 \cdot e^{p_\infty \cdot \overbrace{\nu \cdot \Delta t}^t} \quad (199)$$

Considering that $\nu \cdot \Delta t = t$, (199) says that the error at a time point t is proportional to Δt :

$$\varepsilon(\Delta t, t) \sim \Delta t^1 \quad (200)$$

Eq. (200) illustrates that the **accuracy of the Forward Euler method is of order 1**.

The local error, i.e., from a time point to the next time point, i.e., per Δt , in (199) is of order 2: $\varepsilon(\Delta t) \sim \Delta t^2$.

From (199), it also follows that the error goes to zero over time for a stable system: $\text{Re}\{p_\infty\} < 0 \Rightarrow \lim_{\nu \rightarrow \infty} \varepsilon \rightarrow 0$.

Stability

We assume the test differential equation describes a stable system. That means that the real part of the pole is negative and that the oscillation described by the imaginary part of the pole is decaying towards zero:

$$\text{Re}\{p_\infty\} = \alpha_\infty < 0 \Rightarrow \lim_{\nu \rightarrow \infty} \hat{x}(\nu) \rightarrow 0 \quad (201)$$

To mirror this decaying envelope of the signal x , the following condition must be fulfilled in (194):

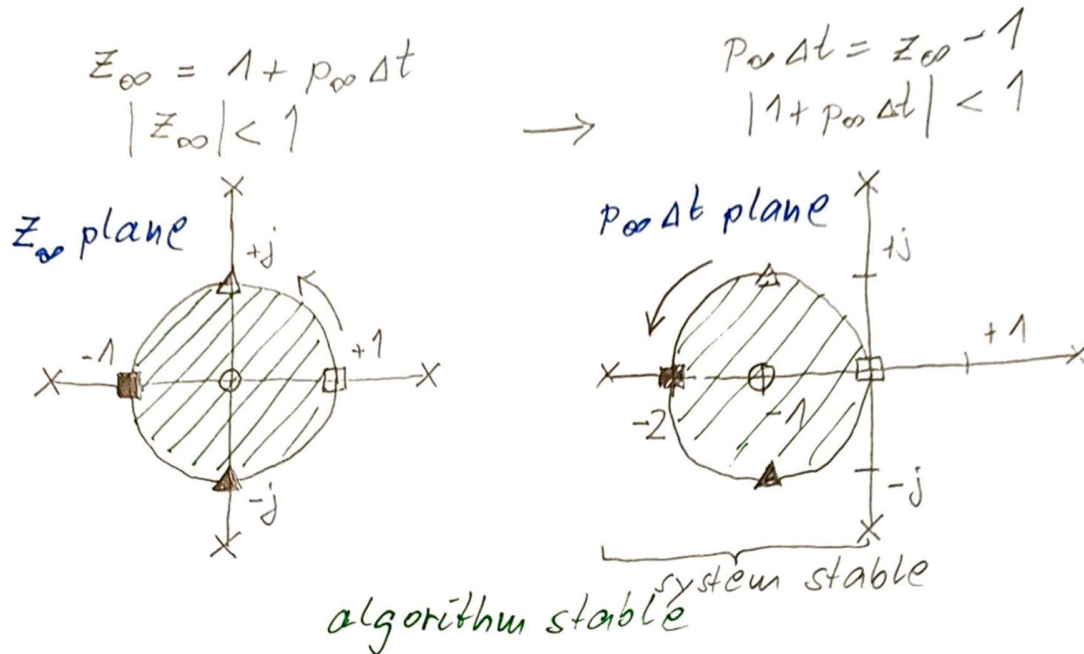
$$|z_\infty| = |1 + p_\infty \cdot \Delta t| < 1 \quad (202)$$

We will visualize the stability by first showing the stability region in the z_∞ plane. The area of $|z_\infty| < 1$ is a circle in the complex plane of z_∞ marked with green hatches on the left of **Figure 49**. We build a conformal map of this area from the z_∞ plane to the $p_\infty \Delta t$ plane. This is done by mapping characteristic points one-to-one and connecting them in the $p_\infty \Delta t$ plane. These points are marked with triangles, boxes, and crosses on the left of **Figure 49**. The corresponding points in the $p_\infty \Delta t$ plane are marked on the right. From the abbreviation of z_∞ in (202), a point in the $p_\infty \Delta t$ plane is a point in the z_∞ plane subtracted by one. The area is obtained by the property that everything on the left left when moving from point to point in a conformal map stays left in both planes. This results in the circular area marked with green hatches on the right of **Figure 49**.

We keep in mind that we assume that the test differential equation describes a stable system, which has a pole with a negative value. This corresponds to the left half plane in the $p_\infty \Delta t$ plane, as Δt is always positive.

The stability of the Forward Euler method is only guaranteed for $|p_\infty \Delta t|$ within the circle on the right side of **Figure 49**. That means that depending on the position of p_∞ in the left half of the $p_\infty \Delta t$ plane, the value of Δt has to be small enough to pull the value of $p_\infty \Delta t$ into this circle. **If Δt is too large, the Forward Euler method is unstable** because it limits the value of the time step in dependence on the system poles.

Figure 49: Visual inspection of the stability of the Forward Euler method on the test differential equation by conformal map.



Asymptotic stability

The asymptotic stability describes the behavior of a numerical integration method for $\Delta t \rightarrow \infty$. In practice, this tells us if we can increase the step size Δt when we observe that the circuit variables asymptotically reach their final values and do not change much anymore.

For **large values of Δt** , (193) can be approximated by:

$$\hat{x}(\nu + 1) \approx p_{\infty} \cdot \Delta t \cdot \hat{x}(\nu) \quad (203)$$

Eq. (203) shows that the **Forward Euler method is asymptotically unstable**. If we increase the step size, the numerically computed integration variable explodes even if the decaying behavior of the test differential equation has been reproduced to a specific time point ν .

6.8.2 Backward Euler method's properties

Structure

A structural interpretation or implementation of the Forward Euler method is obtained by a z-transform of its difference equation (184):

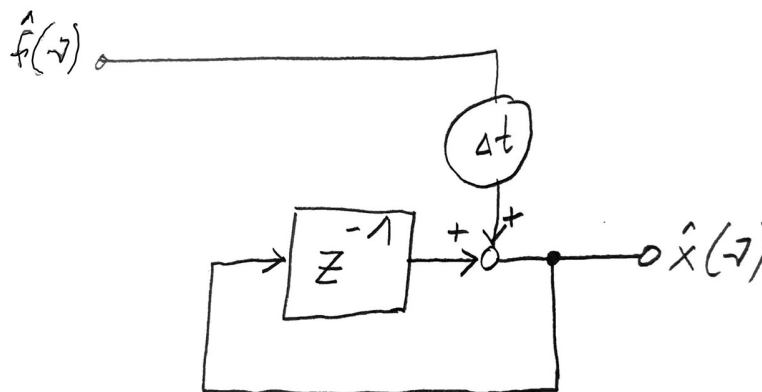
$$(184) \wedge x(0) = 0 \wedge f(0) = 0 \circ \bullet z \cdot \hat{X}(z) = \hat{X}(z) + \Delta t \cdot z \cdot \hat{F}(z) \quad (204)$$

The z-transform (204) can be reformulated in two ways:

$$(204) \Leftrightarrow \hat{X}(z) = \frac{\Delta t \cdot z}{z - 1} \cdot \hat{F}(z) = z^{-1} \cdot \hat{X}(z) + \Delta t \cdot \hat{F}(z) \quad (205)$$

The first expression describes the linear transfer function of a discrete system for numerical integration. The second expression allows us to verify the structure of the Forward Euler numerical integration method in **Figure 50**.

Figure 50: Structure of the Backward Euler numerical integration method.



It represents a hardware/system structure/implementation/interpretation of the Backward Euler method.

State equations

Inserting the discrete state equations given in (175) for \dot{x} in the multivariate version of the difference equations of the Backward Euler method (184) results in the following numerical integration formulas for the Backward Euler method. For linear state equations, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{A} \cdot \hat{\mathbf{x}}(\nu + 1) + \Delta t \cdot \mathbf{b} \cdot u(\nu + 1) \quad (206)$$

$$\Leftrightarrow [\mathbf{I} - \Delta t \cdot \mathbf{A}] \cdot \hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{b} \cdot u(\nu + 1) \quad (207)$$

For the nonlinear case, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}(\nu + 1), \Delta t) \quad (208)$$

The **Backward Euler method is implicit**; the next integration value at time step $\nu+1$ results only implicitly because it is included in the right side of the difference equation. This is illustrated in (208). Eq. (207) clarifies the implicit character of the Backward Euler method for linear state equations. To obtain the values of the next time step $\nu+1$, it is not sufficient to insert values on the right side as for the Forward Euler method. Instead, a linear equation system has to be solved.

Test differential equation

We insert the test differential equation (176) into the difference equation of the Backward Euler method (184) and obtain the following:

$$\hat{x}(\nu + 1) = \hat{x}(\nu) + \Delta t \cdot p_{\infty} \cdot \hat{x}(\nu + 1) = \frac{1}{1 - p_{\infty} \cdot \Delta t} \cdot \hat{x}(\nu) \quad (209)$$

Eq. (209) can also be written as follows:

$$\hat{x}(\nu) = \left(\frac{1}{1 - p_{\infty} \cdot \Delta t} \right)^{\nu} \cdot x(0) \quad (210)$$

Accuracy

The error that the Backward Euler method produces on the test differential equation is obtained as the difference between the numerical solution (210) and the exact solution (177):

$$\varepsilon/x(0) = \varepsilon' = \left(\frac{1}{1 - p_{\infty} \cdot \Delta t} \right)^{\nu} - e^{p_{\infty} \cdot \Delta t \cdot \nu} = e^{-\nu \cdot \ln(1 - p_{\infty} \cdot \Delta t)} - e^{p_{\infty} \cdot \Delta t \cdot \nu} \quad (211)$$

For $|p_{\infty} \cdot \Delta t|$ sufficiently smaller than 1, we can write the following:

$$\ln(1 - p_{\infty} \Delta t) = -p_{\infty} \Delta t - \frac{(p_{\infty} \Delta t)^2}{2} - \dots \approx -p_{\infty} \Delta t \cdot \left(1 + \frac{p_{\infty} \Delta t}{2} \right) \quad (212)$$

We insert (212) into (211) and obtain the following:

$$\varepsilon' \approx e^{\overbrace{p_{\infty} \cdot \nu \cdot \Delta t}^T \cdot \left(1 + \frac{\overbrace{p_{\infty} \cdot \Delta t}^{\delta}}{2} \right)} - e^{p_{\infty} \cdot \nu \cdot \Delta t} = e^{T(1+\delta)} - e^T \quad (213)$$

The abbreviations T , δ , and ε' have been introduced to reduce the writing effort. We compare (213) with (197) and observe that the only difference is in the sign of the definition of δ . The derivations that follow (197) for the Forward Euler method are therefore analogous to the Backward Euler method and lead to the following equation that only differs in the sign from (199):

$$\varepsilon \approx \Delta t \cdot \frac{\overbrace{\nu \cdot \Delta t}^t}{2} \cdot x(0) \cdot p_\infty^2 \cdot e^{p_\infty \cdot \overbrace{\nu \cdot \Delta t}^t} \quad (214)$$

Considering that $\nu \cdot \Delta t = t$, (214) says that the error at a time point t is proportional to Δt :

$$\varepsilon(\Delta t, t) \sim \Delta t^1 \quad (215)$$

Eq. (215) illustrates that the **accuracy of the Backward Euler method is of order 1**.

The local error, i.e., from a time point to the next time point, i.e., per Δt , in (214) is of order 2: $\varepsilon(\Delta t) \sim \Delta t^2$.

From (214), it also follows that the error goes to zero over time for a stable system: $\text{Re}\{p_\infty\} < 0 \Rightarrow \lim_{\nu \rightarrow \infty} \varepsilon \rightarrow 0$.

Stability

We assume the test differential equation describes a stable system. That means that the real part of the pole is negative and that the oscillation described by the imaginary part of the pole is decaying towards zero, as described in (201).

To mirror this decaying envelope of the signal x , the following condition must be fulfilled in (210):

$$|z_\infty| = \left| \frac{1}{1 - p_\infty \cdot \Delta t} \right| < 1 \quad (216)$$

We will follow the same procedure as for the Forward Euler method and visualize the stability by first showing the stability region in the z_∞ plane. The area of $|z_\infty| < 1$ is a circle in the complex plane of z_∞ marked with green hatches on the left of **Figure 51**. We build a conformal map of this area from the z_∞ plane to the $p_\infty \Delta t$ plane. This is done by mapping characteristic points one-to-one and connecting them in the $p_\infty \Delta t$ plane. These points are marked with triangles, boxes, and crosses on the left of **Figure 51**. The corresponding points in the $p_\infty \Delta t$ plane are marked on the right. The area is obtained by the property that everything on the left stays left in a conformal map when moving from point to point in the same direction in both planes. This results in the stability area marked with green hatches on the right of **Figure 51**.

We keep in mind that we assume that the test differential equation describes a stable system, which has a pole with a negative value. This corresponds to the left half plane in the $p_\infty \Delta t$ plane, as Δt is always positive.

The stability of the Backward Euler method is guaranteed for the complete left half plane in the $p_\infty \Delta t$ plane, as shown on the right side of **Figure 51**. There is no restriction on the step length Δt , as was the case for the Forward Euler method. **The Backward Euler method is stable.**

Asymptotic stability

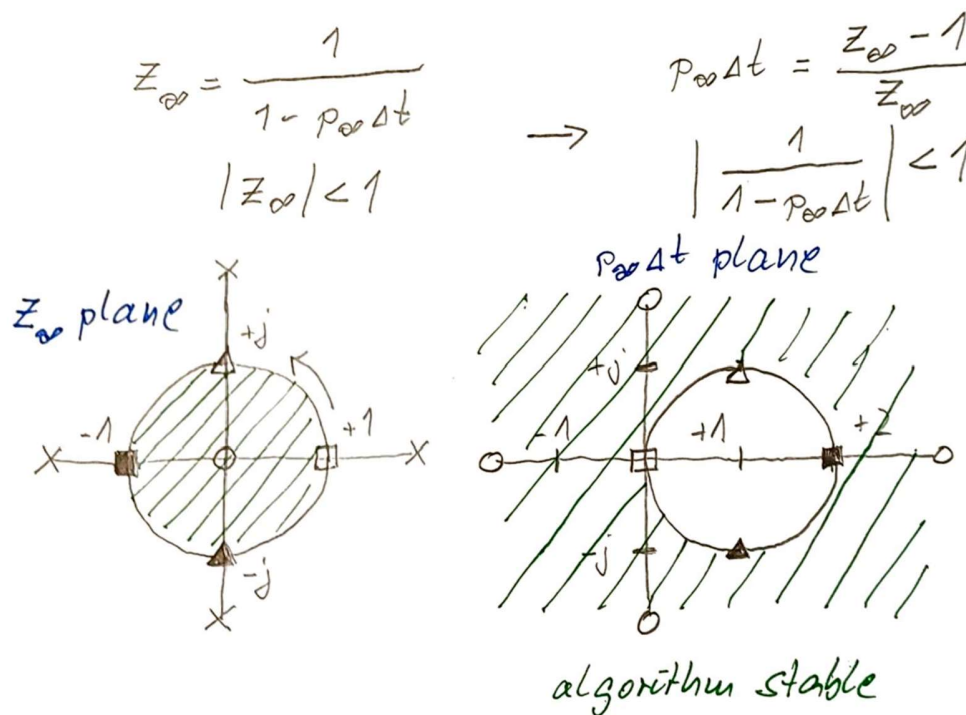
The asymptotic stability describes the behavior of a numerical integration method for $\Delta t \rightarrow \infty$. In practice, this tells us if we can increase the step size Δt when we observe that the circuit variables asymptotically reach their final values and do not change much anymore.

For large values of Δt , (209) can be approximated by:

$$\hat{x}(\nu + 1) \approx -\frac{1}{p_\infty \cdot \Delta t} \cdot \hat{x}(\nu) \Rightarrow \lim_{\Delta t \rightarrow \infty} \hat{x}(\nu + 1) \rightarrow 0 \quad (217)$$

Eq. (217) shows that the **Backward Euler method is asymptotically stable**: it reproduces the decaying behavior of the test differential equation even if the time step length Δt is increased.

Figure 51: Visual inspection of the stability of the Backward Euler method on the test differential equation by conformal map.



6.8.3 Trapezoidal method's properties

A structural interpretation or implementation of the Forward Euler method is obtained by a z-transform of its difference equation (185):

$$(185) \wedge x(0) = 0 \wedge f(0) = 0 \circ \bullet z \cdot \hat{X}(z) = \hat{X}(z) + \frac{\Delta t}{2} \cdot [\hat{F}(z) + z \cdot \hat{F}(z)] \quad (218)$$

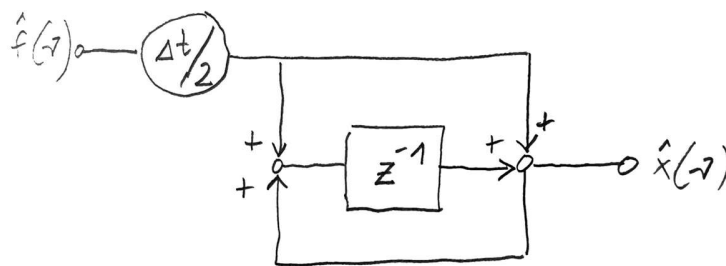
The z-transform (218) can be reformulated in two ways:

$$(218) \Leftrightarrow \hat{X}(z) = \frac{\Delta t}{2} \cdot \frac{z+1}{z-1} \cdot \hat{F}(z) \quad (219)$$

$$= z^{-1} \cdot \hat{X}(z) + \frac{\Delta t}{2} \cdot z^{-1} \cdot \hat{F}(z) + \frac{\Delta t}{2} \cdot \hat{F}(z) \quad (220)$$

The first expression describes the linear transfer function of a discrete system for numerical integration. The second expression allows us to verify the structure of the Trapezoidal numerical integration method in **Figure 52**.

Figure 52: Structure of the Trapezoidal numerical integration method.



It represents a hardware/system structure/implementation/interpretation of the Trapezoidal method.

State equations

Inserting the discrete state equations given in (175) for \dot{x} in the multivariate version of the difference equations of the Trapezoidal method (185) results in the following numerical integration formulas for the Trapezoidal method. For linear state equations, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \frac{\Delta t}{2} \cdot [\mathbf{A} \cdot \hat{\mathbf{x}}(\nu + 1) + \Delta t \cdot \mathbf{b} \cdot u(\nu + 1) + \mathbf{A} \cdot \hat{\mathbf{x}}(\nu) + \Delta t \cdot \mathbf{b} \cdot u(\nu)] \quad (221)$$

$$\Leftrightarrow [\mathbf{I} - \frac{\Delta t}{2} \cdot \mathbf{A}] \cdot \hat{\mathbf{x}}(\nu + 1) = [\mathbf{I} + \frac{\Delta t}{2} \cdot \mathbf{A}] \cdot \hat{\mathbf{x}}(\nu) + \frac{\Delta t}{2} \cdot \mathbf{b} \cdot [u(\nu) + u(\nu + 1)] \quad (222)$$

For the nonlinear case, we obtain the following:

$$\hat{\mathbf{x}}(\nu + 1) = \hat{\mathbf{x}}(\nu) + \frac{\Delta t}{2} \cdot [\mathbf{f}(\hat{\mathbf{x}}(\nu + 1), \Delta t) + \mathbf{f}(\hat{\mathbf{x}}(\nu), \Delta t)] \quad (223)$$

The **Trapezoidal method is implicit**; the next integration value at time step $\nu+1$ results only implicitly because it is included in the right side of the difference equation. This is illustrated in (223). Eq. (222) clarifies the implicit character of the Trapezoidal method for linear state equations. To obtain the values of the next time step $\nu+1$, a linear equation system has to be solved.

Test differential equation

We insert the test differential equation (176) into the difference equation of the Trapezoidal method (185) and obtain the following:

$$\hat{x}(\nu + 1) = \hat{x}(\nu) + \frac{\Delta t \cdot p_{\infty}}{2} \cdot [\hat{x}(\nu + 1) + \hat{x}(\nu)] = \frac{1 + \frac{p_{\infty} \cdot \Delta t}{2}}{1 - \frac{p_{\infty} \cdot \Delta t}{2}} \cdot \hat{x}(\nu) \quad (224)$$

Eq. (224) can also be written as follows:

$$\hat{x}(\nu) = \left(\frac{1 + \frac{p_{\infty} \cdot \Delta t}{2}}{1 - \frac{p_{\infty} \cdot \Delta t}{2}} \right)^{\nu} \cdot x(0) \quad (225)$$

Accuracy

The error that the Trapezoidal method produces on the test differential equation is obtained as the difference between the numerical solution (225) and the exact solution (177):

$$\varepsilon = x(0) \cdot e^{p_{\infty} \cdot \Delta t \cdot \nu} \cdot \underbrace{\left(\frac{e^{-p_{\infty} \cdot \Delta t} \cdot \left(1 + \frac{\Delta t \cdot p_{\infty}}{2}\right)}{\left(1 - \frac{\Delta t \cdot p_{\infty}}{2}\right)} \right)^{\nu} - 1}_T \quad (226)$$

For $|p_{\infty} \cdot \Delta t|$ sufficiently smaller than 1, we develop the term T into series expansions and obtain an approximation:

$$T \approx \left[1 - p_{\infty} \cdot \Delta t + \frac{(p_{\infty} \cdot \Delta t)^2}{2} - \frac{(p_{\infty} \cdot \Delta t)^3}{6} \right] \cdot \left[1 + \frac{p_{\infty} \cdot \Delta t}{2} \right] \cdot \left[1 + \frac{p_{\infty} \cdot \Delta t}{2} + \frac{(p_{\infty} \cdot \Delta t)^2}{4} + \frac{(p_{\infty} \cdot \Delta t)^3}{8} \right] \quad (227)$$

$$T \approx \left[1 + \frac{p_{\infty} \cdot \Delta t}{2} - p_{\infty} \cdot \Delta t - \frac{(p_{\infty} \cdot \Delta t)^2}{2} + \frac{(p_{\infty} \cdot \Delta t)^2}{2} + \frac{(p_{\infty} \cdot \Delta t)^3}{4} - \frac{(p_{\infty} \cdot \Delta t)^3}{6} \right] \cdot \left[1 + \frac{p_{\infty} \cdot \Delta t}{2} + \frac{(p_{\infty} \cdot \Delta t)^2}{4} + \frac{(p_{\infty} \cdot \Delta t)^3}{8} \right] \quad (228)$$

$$T \approx \left[1 - \frac{p_{\infty} \cdot \Delta t}{2} + \frac{(p_{\infty} \cdot \Delta t)^3}{12} \right] \cdot \left[1 + \frac{p_{\infty} \cdot \Delta t}{2} + \frac{(p_{\infty} \cdot \Delta t)^2}{4} + \frac{(p_{\infty} \cdot \Delta t)^3}{8} \right] \quad (229)$$

$$T \approx 1 + \frac{p_\infty \cdot \Delta t}{2} + \frac{(p_\infty \cdot \Delta t)^2}{4} + \frac{(p_\infty \cdot \Delta t)^3}{8} - \frac{p_\infty \cdot \Delta t}{2} - \frac{(p_\infty \cdot \Delta t)^2}{4} - \frac{(p_\infty \cdot \Delta t)^3}{8} + \frac{(p_\infty \cdot \Delta t)^3}{12} \quad (230)$$

$$\approx 1 + \frac{(p_\infty \cdot \Delta t)^3}{12} \quad (231)$$

Inserting (231) in the parentheses part of (226) results in

$$\left(1 + \frac{(p_\infty \cdot \Delta t)^3}{12}\right)^\nu - 1 \approx \nu \frac{(p_\infty \cdot \Delta t)^3}{12} \quad (232)$$

Inserting (232) in (226) finally leads to the following formula:

$$\varepsilon \approx \Delta t^2 \cdot \overbrace{\frac{\nu \cdot \Delta t}{12}}^t \cdot x(0) \cdot p_\infty^3 \cdot e^{p_\infty \cdot \overbrace{\nu \cdot \Delta t}^t} \quad (233)$$

Considering that $\nu \cdot \Delta t = t$, (214) says that the error at a time point t is proportional to Δt^2 :

$$\varepsilon(\Delta t, t) \sim \Delta t^2 \quad (234)$$

Eq. (234) illustrates that the **accuracy of the Trapezoidal method is of order 2**.

From (233), it also follows that the error goes to zero over time for a stable system: $\text{Re}\{p_\infty\} < 0 \Rightarrow \lim_{\nu \rightarrow \infty} \varepsilon \rightarrow 0$.

Stability

We assume the test differential equation describes a stable system. That means that the real part of the pole is negative and that the oscillation described by the imaginary part of the pole is decaying towards zero, as described in (201).

To mirror this decaying envelope of the signal x , the following condition must be fulfilled in (225):

$$|z_\infty| = \left| \frac{1 + \frac{p_\infty \cdot \Delta t}{2}}{1 - \frac{p_\infty \cdot \Delta t}{2}} \right| < 1 \quad (235)$$

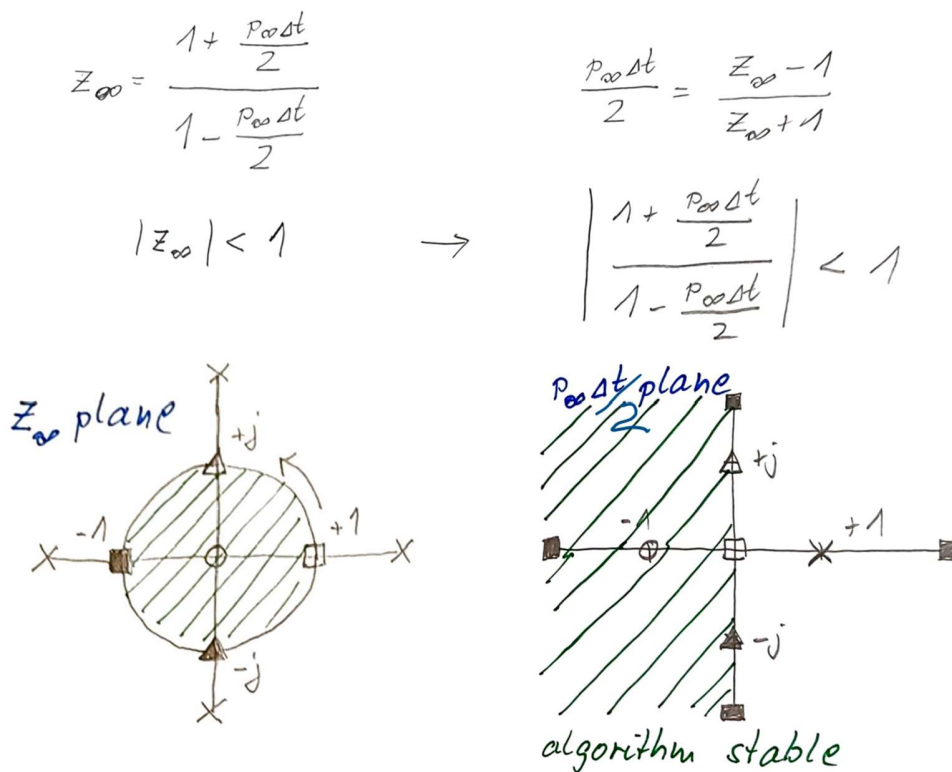
We will follow the same procedure as for the Forward Euler method and visualize the stability by first showing the stability region in the z_∞ plane. The area of $|z_\infty| < 1$ is a circle in the complex plane of z_∞ marked with green hatches on the left of **Figure 53**. We build a conformal map of this area from the z_∞ plane to the $p_\infty \Delta t / 2$ plane. This is done by mapping characteristic points one-to-one and connecting them in the $p_\infty \Delta t / 2$ plane. These points are marked with triangles, boxes, and crosses on the left of **Figure 53**. The corresponding points in the $p_\infty \Delta t / 2$ plane are marked on the right. The area is obtained by the property that everything on the left stays left in a conformal map

when moving from point to point in the same direction in both planes. This results in the area marked with green hatches on the right of **Figure 53**.

We keep in mind that we assume that the test differential equation describes a stable system, which has a pole with a negative value. This corresponds to the left half plane in the $p_{\infty}\Delta t/2$ plane, as Δt is always positive.

The stability of the Backward Euler method is guaranteed for the complete left half plane in the $p_{\infty}\Delta t/2$ plane, as shown on the right side of **Figure 53**. There is no restriction on the step length Δt , as was the case for the Forward Euler method. **The Trapezoidal method is stable.**

Figure 53: Visual inspection of the stability of the Trapezoidal method on the test differential equation by conformal map.



Asymptotic stability

The asymptotic stability describes the behavior of a numerical integration method for $\Delta t \rightarrow \infty$. In practice, this tells us if we can increase the step size Δt when we observe that the circuit variables asymptotically reach their final values and do not change much anymore.

For **large values of Δt** , (224) can be approximated by:

$$\hat{x}(\nu + 1) \approx -\hat{x}(\nu) \Rightarrow \lim_{\Delta t \rightarrow \infty} \hat{x}(\nu + 1) \neq 0 \quad (236)$$

Eq. (236) shows that the **Trapezoidal method is asymptotically unstable**: it oscillates around the recent value if the time step length Δt is increased, instead of continuing to decrease.

6.8.4 Second-order Gear method's properties

A structural interpretation or implementation of the second-order Gear method is obtained by a z-transform of its difference equation (187):

$$(187) \wedge x(0) = x(1) = 0 \wedge f(0) = f(1) = 0$$

$$z^2 \cdot \hat{X}(z) = \frac{4}{3}z \cdot \hat{X}(z) - \frac{1}{3}\hat{X}(z) + \frac{2\Delta t}{3} \cdot z^2 \cdot \hat{F}(z) \quad (237)$$

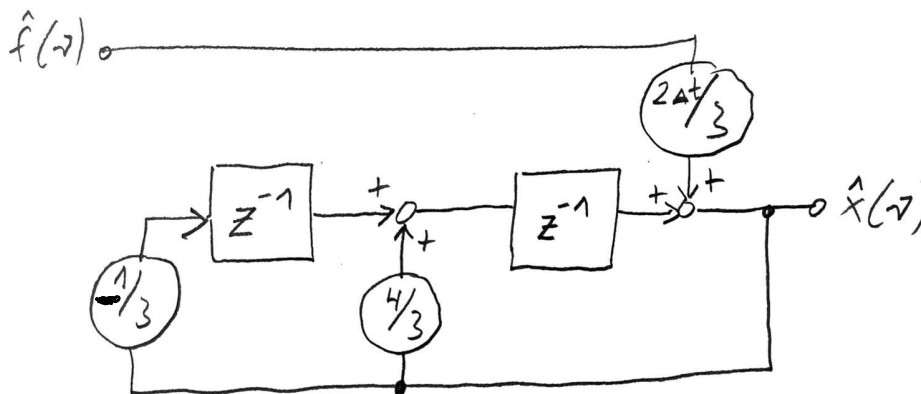
The z-transform (237) can be reformulated in two ways

$$(237) \Leftrightarrow \hat{X}(z) = \frac{\frac{2\Delta t}{3} \cdot z^2}{z^2 - \frac{4}{3}z + \frac{1}{3}} \cdot \hat{F}(z) \quad (238)$$

$$= \frac{4}{3}z^{-1} \cdot \hat{X}(z) - \frac{1}{3}z^{-2} \cdot \hat{X}(z) + \frac{2\Delta t}{3} \cdot \hat{F}(z) \quad (239)$$

The first expression describes the linear transfer function of a discrete system for numerical integration. The second expression allows us to verify the structure of the second-order Gear numerical integration method in **Figure 54**.

Figure 54: Structure of the second-order Gear numerical integration method.



It represents a hardware/system structure/implementation/interpretation of the second-order Gear method.

State equations

Inserting the discrete state equations given in (175) for \dot{x} in the multivariate version of the difference equations of the second-order Gear method (187) results in the following numerical integration formulas for the second-order Gear method. For linear state equations, we obtain the following:

$$\left[\mathbf{I} - \frac{2\Delta t}{3} \cdot \mathbf{A} \right] \cdot \hat{\mathbf{x}}(\nu + 2) = \frac{4}{3} \hat{\mathbf{x}}(\nu + 1) - \frac{1}{3} \hat{\mathbf{x}}(\nu) + \frac{2\Delta t}{3} \cdot \mathbf{b} \cdot u(\nu + 2) \quad (240)$$

The **second-order Gear method is implicit.**

Test differential equation

Using the test differential equation (176), it can be illustrated that the second-order Gear method has an accuracy of order 2, is numerically stable for a stable system, and is asymptotically stable for large time steps.

6.8.5 Summary

Overall, the properties of the mentioned four numerical integration methods can be summarized in the following table.

	Explicit/implicit	m-step	Error of order k	Stable algorithm for a stable system	Stable for $\Delta t \rightarrow \infty$
Forward Euler	e	1	1	no	no ("explodes")
Backward Euler	i	1	1	yes	yes
Trapezoidal	i	1	2	yes	no ("toggles")
Second-order Gear	i	2	2	yes	yes

6.9 Heun method

The Heun method can serve as an example of a so-called predictor-corrector method.

The Forward Euler method is used as predictor $x_P(\nu+1)$ of the integration value in the next time step:

$$x_P(\nu + 1) = x(\nu) + \Delta t \cdot \dot{x}(\nu) \quad (241)$$

Then, we calculate the differential (function to be integrated) at the next time step:

$$\dot{x}_P(\nu + 1) = f(x_P(\nu + 1), \nu + 1) \quad (242)$$

The average of the predicted differential at time step $\nu+1$ and the differential at time step ν are averaged to provide a corrector \dot{x}_C :

$$\dot{x}_C(\nu, \nu + 1) = \frac{1}{2} (\dot{x}(\nu) + \dot{x}_P(\nu + 1)) \quad (243)$$

This finite differential \dot{x}_C resembles the Trapezoidal method. However, it has a lower computational effort because (242) is an explicit formula. The next time point then is:

$$x(\nu + 1) = x(\nu) + \Delta t \cdot \dot{x}_C(\nu, \nu + 1) \quad (244)$$

7. The discretized and linearized nodal voltage system in time step $v+1$ and iteration step $\mu+1$

We will now set up the nodal voltage system for a transient simulation of a circuit with energy-storing elements and nonlinear elements. This will include discretization with a numerical integration method and linearization. It will lead to two computational loops: one over the discretized time v and one over the nonlinear iteration μ .

The outer loop will be over the time. In each time step, an inner iterative loop over the nonlinearities will be opened.

We will concentrate on inductors and capacitors, and on the Backward Euler numerical integration method. Please note that there is one Taylor series and Newton approach for nonlinear equations/elements, but many (here: four) numerical integration methods for the differential equations of energy-storing elements.

We will discretize and linearize edge-wise/element-wise and set up nodal voltage equations in time step $v+1$ and iteration step $\mu+1$. Afterward, we will present a circuit-equivalent interpretation of the numerical integration method and use it to set up the discretized and linearized nodal equations immediately.

7.1 Discretizing and linearizing the differential and nonlinear nodal system equations

The Kirchhoff current law, the branch constitutive equations, and the Kirchhoff voltage law will be set up separately for edges with (in part nonlinear) resistive elements, capacitive elements, and inductive elements:

$$(KCL) \mathbf{A} \cdot \mathbf{i} = [\mathbf{A}_R \ \mathbf{A}_C \ \mathbf{A}_L] \cdot \begin{bmatrix} \mathbf{i}_R \\ \mathbf{i}_C \\ \mathbf{i}_L \end{bmatrix} = \mathbf{A}_R \cdot \mathbf{i}_R + \mathbf{A}_C \cdot \mathbf{i}_C + \mathbf{A}_L \cdot \mathbf{i}_L = \mathbf{0} \quad (245)$$

$$(BCE_R) \mathbf{i}_R - \mathbf{I}_{0R} = \mathbf{g}(\mathbf{u}_R - \mathbf{U}_{0R}) \quad (246)$$

$$(BCE_C) \mathbf{i}_C = \mathbf{C} \cdot \dot{\mathbf{u}}_C \quad (247)$$

$$(BCE_L) \mathbf{u}_L = \mathbf{L} \cdot \dot{\mathbf{i}}_L \quad (248)$$

$$(KVL) \mathbf{u} = \begin{bmatrix} \mathbf{u}_R \\ \mathbf{u}_C \\ \mathbf{u}_L \end{bmatrix} = \mathbf{A}^T \cdot \mathbf{u}_n = \begin{bmatrix} \mathbf{A}_R^T \\ \mathbf{A}_C^T \\ \mathbf{A}_L^T \end{bmatrix} \cdot \mathbf{u}_n \Leftrightarrow \begin{aligned} \mathbf{u}_R &= \mathbf{A}_R^T \cdot \mathbf{u}_n \\ \mathbf{u}_C &= \mathbf{A}_C^T \cdot \mathbf{u}_n \\ \mathbf{u}_L &= \mathbf{A}_L^T \cdot \mathbf{u}_n \end{aligned} \quad (249)$$

The branch constitutive equations for capacitors and inductors are the corresponding differential equations. For simplicity, we have assumed that there are no constant sources in branches with capacitive and inductive elements. \mathbf{C} and \mathbf{L} are diagonal matrices. Controlled sources and Z-branches can be treated as described before.

Discretization (Backward Euler method)

Eqs. (247) and (248) are differential equations and subject to discretization. The Backward Euler difference equation (184) is applied with x being the variable appearing as time derivative ("dot") in (247) and (248). For

(247), this leads intermediately to $\mathbf{u}_C(\nu + 1) = \mathbf{u}_C(\nu) + \Delta t \cdot \mathbf{C}^{-1} \cdot \mathbf{i}_C(\nu + 1)$ and the following two difference equations for the branch constitutive equations (247) and (248):

$$\widehat{(\text{BCE}_C)} \quad \mathbf{i}_C(\nu + 1) = \frac{1}{\Delta t} \cdot \mathbf{C} \cdot \mathbf{u}_C(\nu + 1) - \frac{1}{\Delta t} \cdot \mathbf{C} \cdot \mathbf{u}_C(\nu) \quad (250)$$

$$\widehat{(\text{BCE}_L)} \quad \mathbf{i}_L(\nu + 1) = \mathbf{i}_L(\nu) + \Delta t \cdot \mathbf{L}^{-1} \cdot \mathbf{u}_L(\nu + 1) \quad (251)$$

Linearization

The nonlinear elements are linearized (i.e., Taylor series of first order) in the actual time step $\nu+1$. This leads to the following linearized version of the branch constitutive equation (246):

$$\overline{(\text{BCE}_R)} \quad \mathbf{i}_R^{(\mu+1)}(\nu + 1) = \mathbf{i}_R^{(\mu)}(\nu + 1) \quad (252)$$

$$+ \underbrace{\left. \frac{\partial \mathbf{g}}{\partial \mathbf{u}_R^T} \right|_{(\nu + 1)}}_{\mathbf{Y}_R^{(\mu)}(\nu+1)} \cdot \left(\mathbf{u}_R^{(\mu+1)}(\nu + 1) - \mathbf{u}_R^{(\mu)}(\nu + 1) \right)$$

$$\text{with} \quad \mathbf{i}_R^{(\mu)}(\nu + 1) = \mathbf{I}_{0R} + \mathbf{g}(\mathbf{u}_R^{(\mu)}(\nu + 1) - \mathbf{U}_{0R}) \quad (253)$$

Discretized and linearized nodal equation system

As done before, the equations will be inserted into each other:

$$(\text{KVL}) \longrightarrow \overline{(\text{BCE}_R)}, \widehat{(\text{BCE}_C)}, \widehat{(\text{BCE}_L)} \longrightarrow (\text{KCL}) \quad (254)$$

This leads to the following equation. Please note that the discretized branch constitutive equations (250) and (251) are extended with the current iteration step $\mu+1$ to be computed at the present time step $\nu+1$ to be computed. The terms for the previous time step ν are constant, as the nonlinear iteration of the past step ν has been finished, and these terms are constant in the present time step $\nu+1$.

$$\begin{aligned} & \mathbf{A}_R \cdot \mathbf{i}_R^{(\mu)}(\nu + 1) + \underbrace{\mathbf{A}_R \cdot \mathbf{Y}_R^{(\mu)}(\nu + 1) \cdot \mathbf{A}_R^T}_{\mathbf{Y}_n^{(\mu)}(\nu+1)} \cdot \left(\mathbf{u}_n^{(\mu+1)}(\nu + 1) - \mathbf{u}_n^{(\mu)}(\nu + 1) \right) \\ & + \underbrace{\frac{1}{\Delta t} \cdot \mathbf{A}_C \cdot \mathbf{C} \cdot \mathbf{A}_C^T}_{\mathbf{Y}_{nC}} \cdot \left(\mathbf{u}_n^{(\mu+1)}(\nu + 1) - \mathbf{u}_n(\nu) \right) \\ & + \mathbf{A}_L \cdot \mathbf{i}_L(\nu) + \underbrace{\Delta t \cdot \mathbf{A}_L \cdot \mathbf{L}^{-1} \cdot \mathbf{A}_L^T}_{\mathbf{Y}_{nL}} \cdot \mathbf{u}_n^{(\mu+1)}(\nu + 1) = \mathbf{0} \end{aligned} \quad (255)$$

Sorting the left side and right of (255) and using the abbreviations introduced in (255) leads to the Backward Euler-based discretized and linearized nodal equation system:

$$\begin{aligned} & \left(\mathbf{Y}_n^{(\mu)}(\nu + 1) + \mathbf{Y}_{nC} + \mathbf{Y}_{nL} \right) \cdot \mathbf{u}_n^{(\mu+1)}(\nu + 1) = \\ & \mathbf{Y}_n^{(\mu)}(\nu + 1) \cdot \mathbf{u}_n^{(\mu)}(\nu + 1) - \mathbf{A}_R \cdot \mathbf{i}_R^{(\mu)}(\nu + 1) + \mathbf{Y}_{nC} \cdot \mathbf{u}_n(\nu) - \mathbf{A}_L \cdot \mathbf{i}_L(\nu) \end{aligned} \quad (256)$$

We have said that the transient simulation walks from discrete time step to step. Within each time step, an iteration loop is opened to cope with the nonlinear elements. Time step $\nu+1$ represents the current time step for which we want to compute the node voltages of the circuit. Within that time step $\nu+1$, in turn, the current iteration step for which we want to compute the node voltages is denoted by $\mu+1$. Hence, the unknown system variables in (256) are $\mathbf{u}_n^{(\mu+1)}(\nu + 1)$. Time step ν represents the recent time step; values from that time step have been computed and define constants on the right side of (256). Then, there are terms from the current time step $\nu+1$, but from the last iteration step μ in the present time step. These terms appear both on the right side and in the system matrix on the left side and need to be updated in each iteration step within the current time step. The overall transient simulation flow is sketched in the following.

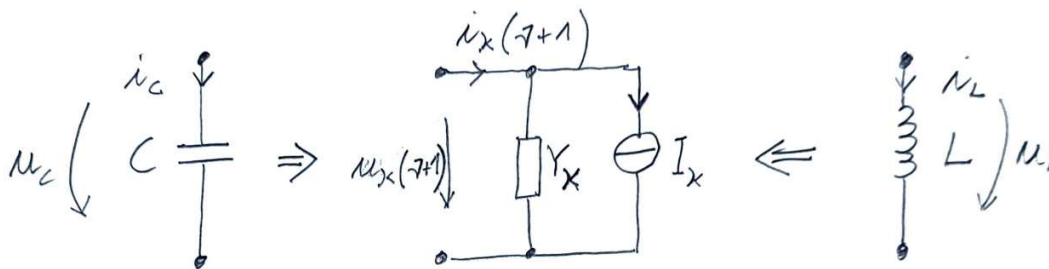
7.2 Transient simulation flow

Compute initial values $\mathbf{u}_n(0)$, $\mathbf{i}_L(0)$, $\mathbf{i}_C(0)$	
Set $\nu:=0$	
Repeat	Set $\mu:=0$ (start iteration loop)
	Repeat
	If $\mu > \mu_{\max}$ then ERROR, ABORT (maximum number of iterations reached without convergence)
	Compute $\mathbf{i}_R^{(\mu)}(\nu + 1)$ according to (253)
	Compute $\mathbf{Y}_R^{(\mu)}(\nu + 1)$ according to (252)
	Update the entries in the matrix and right side of (256) using abbreviations in (255)
	Solve the linear equation system (256)
	Set $\mu:=\mu+1$ (switch to next iteration step)
	Until
	$ \mathbf{u}_n^{(\mu+1)} - \mathbf{u}_n^{(\mu)} < \epsilon$
Set $\mathbf{u}_n(\nu + 1) := \mathbf{u}_n^{(\mu)}(\nu + 1)$ (node voltages in actual time step computed)	
Compute $\mathbf{i}_L(\nu+1)$, $\mathbf{i}_C(\nu+1)$ according to (250) and (251)	
Set $\nu:=\nu+1$ (switch to next time step)	
Until	$\nu > \nu_{\max}$ (last time step computed)

7.3 Discretization – circuit equivalent

As we exercised in the case of linearization, we will now present an interpretation of the difference equations of capacitance and inductance with a circuit equivalent. We will discover that the discretized model can be interpreted by a standard branch κ consisting of an admittance and current source in parallel connection, as shown in **Figure 55**, or, analogously, as a series connection of impedance and voltage source. **Figure 55** shows the current time step $\nu+1$ to be computed for a one-step integration method. For the two-step second-order Gear method, it would be $\nu+2$.

Figure 55: Equivalent circuit of capacitance and inductance for one-step numerical integration method.



In the following, we will derive the equivalent circuit elements in **Figure 55** for the Trapezoidal method. Antreich's original lecture notes give an overview of all treated numerical integration methods and for a series connection.

Discretized inductance – Trapezoidal method

The following steps lead to the equivalent circuit interpretation of a numerical integration method.

- 1) Note down the difference equation of the selected numerical integration method, here, the Trapezoidal method.

$$x(\nu + 1) = x(\nu) + \frac{\Delta t}{2} \cdot (f(\nu) + f(\nu + 1)) , \text{ with } \dot{x}(t) = f(x(t), t) \quad (257)$$

- 2) Note down the differential equation of the circuit element, here, an inductance.

$$u_L = L \cdot \dot{i}_L \quad (258)$$

- 3) Combine (257) and (258). x is the variable that appears with its time derivative in (258).

$$i_L(\nu + 1) = i_L(\nu) + \frac{\Delta t}{2L} \cdot (u_L(\nu) + u_L(\nu + 1)) \quad (259)$$

- 4) (Re)formulate (259) with i_κ in dependence of u_κ for an equivalent circuit as a parallel connection of admittance and current source as shown in **Figure 55** (or, u_κ in dependence of i_κ for an equivalent circuit as the serial connection of impedance and voltage source). Order and group the terms according to indexes.

$$i_L(\nu + 1) = \frac{\Delta t}{2L} \cdot u_L(\nu + 1) + \left(i_L(\nu) + \frac{\Delta t}{2L} \cdot u_L(\nu) \right) \quad (260)$$

5) Interpret the equivalent circuit's elements according to **Figure 55**.

$$Y_\kappa = \frac{\Delta t}{2L} \quad (261)$$

$$I_\kappa(\nu) = i_L(\nu) + \frac{\Delta t}{2L} \cdot u_L(\nu) \quad (262)$$

Discretized capacitance – Trapezoidal method

For the capacitance to be discretized with the Trapezoidal method, step 1) is the same as above. Steps 2) to 5) are as follows.

$$2) \quad i_C = C \cdot \dot{u}_C \quad (263)$$

$$3) \quad u_C(\nu + 1) = u_C(\nu) + \frac{\Delta t}{2C} \cdot (i_C(\nu) + i_C(\nu + 1)) \quad (264)$$

$$4) \quad i_C(\nu + 1) = \frac{2C}{\Delta t} \cdot u_C(\nu + 1) - i_C(\nu) - \frac{2C}{\Delta t} \cdot u_C(\nu) \quad (265)$$

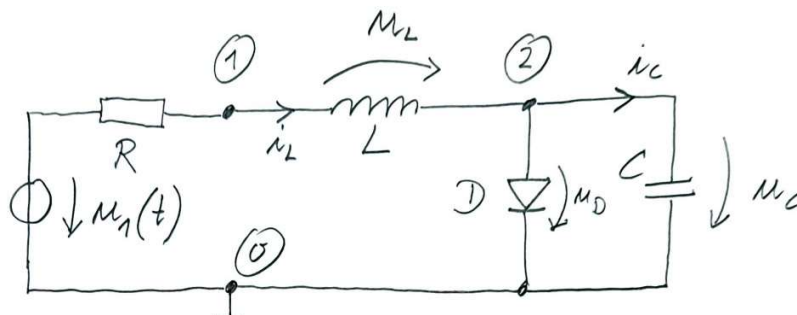
$$5) \quad Y_\kappa = \frac{2C}{\Delta t} \quad (266)$$

$$I_\kappa(\nu) = -i_C(\nu) - \frac{2C}{\Delta t} \cdot u_C(\nu) \quad (267)$$

7.4 Example for setting up a discretized and linearized nodal equation system

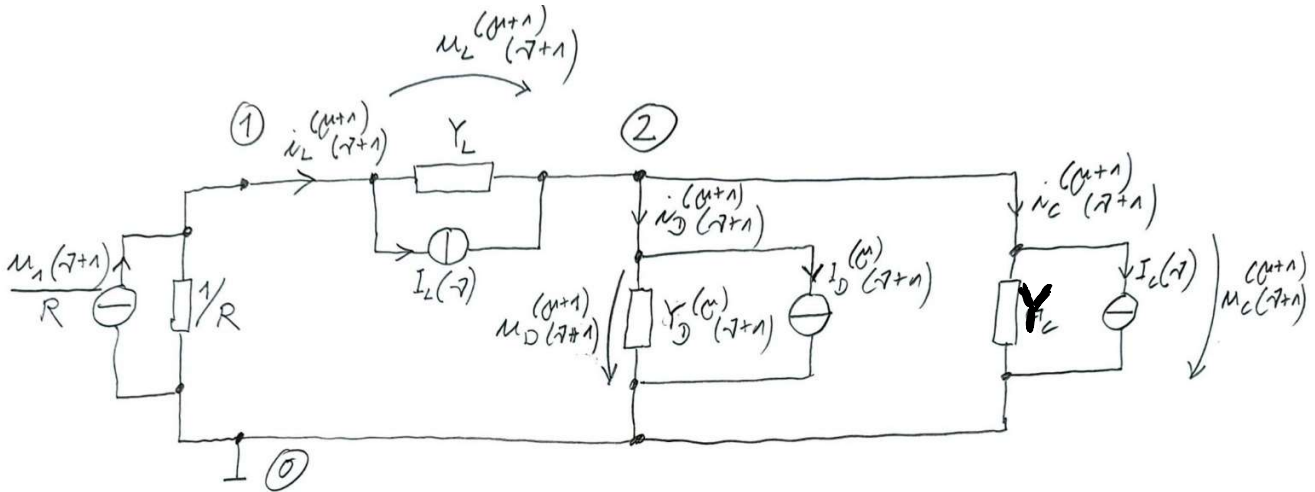
The circuit in **Figure 56** will serve to illustrate setting up the discretized and linearized nodal system. We select the Trapezoidal method for numerical integration. For the nonlinear diode, we will assume the current equation in **Figure 30**.

Figure 56: Circuit with nonlinear and energy-storing elements to be time-discretized and linearized.



This means that we can use **Figure 55** and (261), (262), (266) and (267) for the discretized equivalent circuits of the inductor and capacitor, and **Figure 37** and (150)-(151) for the linearized equivalent circuit of the diode. We obtain the discretized and linearized equivalent circuit in **Figure 57**.

Figure 57: Time-discretized and linearized circuit of the circuit in Figure 56.



The following equations are adapted from the ones mentioned above and describe entirely the discretized and linearized equivalent circuit.

Inductor:

$$Y_L = \frac{\Delta t}{2L} \quad (268)$$

$$I_L(\nu) = i_L(\nu) + \frac{\Delta t}{2L} \cdot u_L(\nu) \quad (269)$$

$$u_L = u_{n1} - u_{n2} \quad (270)$$

Capacitor:

$$Y_C = \frac{2C}{\Delta t} \quad (271)$$

$$I_C(\nu) = -i_C(\nu) - \frac{2C}{\Delta t} \cdot u_C(\nu) \quad (272)$$

$$u_C = u_{n2} \quad (273)$$

Diode:

$$Y_D^{(\mu)}(\nu + 1) = \frac{I_S}{U_T} \cdot e^{\frac{u_D^{(\mu)}(\nu+1)}{U_T}} \quad (274)$$

$$I_D^{(\mu)}(\nu + 1) = i_D^{(\mu)}(\nu + 1) - Y_D^{(\mu)}(\nu + 1) \cdot u_D^{(\mu)}(\nu + 1) \quad (275)$$

$$i_D^{(\mu)}(\nu + 1) = I_S \cdot (e^{u_D^{(\mu)}(\nu+1)/U_T} - 1) \quad (276)$$

$$u_D = u_{n2} \quad (277)$$

For the discretized and linearized circuit in **Figure 57**, the nodal equation system can be set up according to the rules in Sec. 2.4. We obtain the following nodal equation system.

Discretized and linearized nodal equation system:

$$\begin{aligned} \begin{bmatrix} \frac{1}{R} + Y_L & -Y_L \\ -Y_L & Y_L + Y_C + Y_D^{(\mu)}(\nu + 1) \end{bmatrix} \cdot \begin{bmatrix} u_{n1} \\ u_{n2} \end{bmatrix}^{(\nu+1)} \\ = \begin{bmatrix} \frac{u_1(\nu+1)}{R} - I_L(\nu) \\ I_L(\nu) - I_D^{(\mu)}(\nu + 1) - I_C(\nu) \end{bmatrix} \end{aligned} \quad (278)$$

Eq. (278) and the equations (268)-(277) for its entries are the basis for a transient simulation of the circuit in **Figure 56** following the procedure in Sec. 7.2.

7.5 Nonlinearity and energy storage in the same element

Nonlinearity and differentials may appear in the same element. We will illustrate how time discretization and linearization can be done, for example, for a nonlinear capacitor and a nonlinear inductor. The essence of the approach is a variable in which both the nonlinearity and the differential “meet”. This variable will be the charge for the capacitor and the flux for the inductor in the following.

7.5.1 Nonlinear capacitor

The differential equation for a capacitor with a charge that is nonlinearly dependent on the voltage is as follows:

$$i = \dot{q} = \frac{dq}{du} \cdot \frac{du}{dt} = \frac{dq}{du} \cdot \frac{du}{dt} = C(u) \cdot \dot{u} \quad (279)$$

The nonlinearity is marked in green in (279). The differential part is marked in red. Both “meet” at the charge q . The nonlinear function could be, for example, as follows:

$$q(u) = \frac{2Q_0}{\pi} \arctan(u/U_0) \quad (280)$$

First, the charge q is discretized. We will use the Backward Euler method:

$$q(\nu + 1) = q(\nu) + \Delta t \cdot i(\nu + 1) \quad (281)$$

Eq. (281) is reformulated with the current on the left side:

$$i(\nu + 1) = \frac{1}{\Delta t} \cdot q(\nu + 1) - \frac{1}{\Delta t} \cdot q(\nu) \quad (282)$$

Now, we have to linearize the charge in the actual time step $\nu+1$ to open the iteration loop over the nonlinearity of the charge. This nonlinearity is formulated in dependence on the voltage, which brings back the connection between branch voltage and branch current and thus leads to the nodal voltage system. Linearizing q in time step $\nu+1$ leads to the following:

$$q^{(\mu+1)}(\nu+1) = q^{(\mu)}(\nu+1) + \underbrace{\frac{\partial q}{\partial u} \Big|_{(\nu+1)}^{(\mu)}}_{C^{(\mu)}(\nu+1)} \cdot (u^{(\mu+1)}(\nu+1) - u^{(\mu)}(\nu+1)) \quad (283)$$

In our example, (280), the nonlinear capacitance would be:

$$C^{(\mu)}(\nu+1) = \frac{2Q_0}{\pi U_0} \frac{1}{1 + \left(\frac{u^{(\mu)}(\nu+1)}{U_0}\right)^2} \quad (284)$$

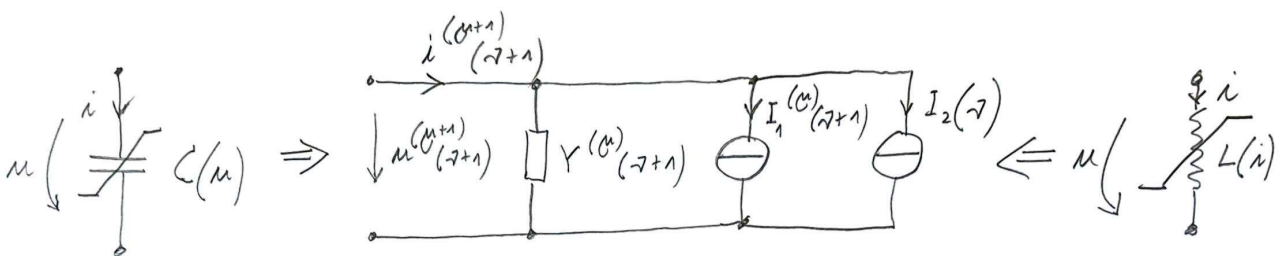
Discretization and linearization are combined by inserting (283) in (282) as marked in blue:

$$i^{(\mu+1)}(\nu+1) = \underbrace{\frac{C^{(\mu)}(\nu+1)}{\Delta t}}_{Y^{(\mu)}(\nu+1)} \cdot u^{(\mu+1)}(\nu+1) + \underbrace{\frac{1}{\Delta t} \cdot q^{(\mu)}(\nu+1) - \frac{C^{(\mu)}(\nu+1)}{\Delta t} \cdot u^{(\mu)}(\nu+1)}_{I_1^{(\mu)}(\nu+1)} - \underbrace{\frac{1}{\Delta t} \cdot q(\nu)}_{I_2(\nu)} \quad (285)$$

In (285), the terms have been ordered in groups of indexes. These groups can be interpreted as the components of an equivalent circuit of the discretized and linearized capacitor shown in

Figure 58.

Figure 58: Equivalent circuit of nonlinear capacitance and inductance for one-step numerical integration method.



7.5.2 Nonlinear inductor

The differential equation for an inductor with a flux that is nonlinearly dependent on the current is as follows:

$$u = \dot{\Phi} = \frac{d\Phi}{di} \cdot \frac{di}{dt} = \frac{d\Phi}{di} \cdot \frac{di}{dt} = L(i) \cdot \dot{i} \quad (286)$$

The nonlinearity is marked in green in (286). The differential part is marked in red. Both “meet” at the flux Φ . The nonlinear function could be, for example, as follows:

$$\Phi(i) = \Phi_0 \cdot \left(\frac{i}{I_0}\right)^{\frac{1}{3}} \quad (287)$$

First, the flux Φ is discretized. We will use the Backward Euler method:

$$\Phi(\nu + 1) = \Phi(\nu) + \Delta t \cdot u(\nu + 1) \quad (288)$$

Eq. (288) is reformulated with the voltage on the left side:

$$u(\nu + 1) = \frac{1}{\Delta t} \cdot \Phi(\nu + 1) - \frac{1}{\Delta t} \cdot \Phi(\nu) \quad (289)$$

Now, we have to linearize the flux in the actual time step $\nu+1$ to open the iteration loop over the nonlinearity of the flux. This nonlinearity is formulated in dependence on the current, which brings back the connection between branch voltage and branch current and thus leads to the nodal voltage system. Linearizing Φ in time step $\nu+1$ leads to the following:

$$\Phi^{(\mu+1)}(\nu + 1) = \Phi^{(\mu)}(\nu + 1) + \underbrace{\frac{\partial \Phi}{\partial i} \Big|_{(\nu + 1)}^{(\mu)}}_{L^{(\mu)}(\nu+1)} \cdot (i^{(\mu+1)}(\nu + 1) - i^{(\mu)}(\nu + 1)) \quad (290)$$

In our example, (287), the nonlinear inductance would be:

$$L^{(\mu)}(\nu + 1) = \frac{\Phi_0}{3I_0} \cdot \left(\frac{i^{(\mu)}(\nu + 1)}{I_0}\right)^{-\frac{2}{3}} \quad (291)$$

Discretization and linearization are combined by inserting (290) in (289) as marked in blue:

$$u^{(\mu+1)}(\nu + 1) = \underbrace{\frac{L^{(\mu)}(\nu + 1)}{\Delta t}}_{Z^{(\mu)}(\nu+1)} \cdot i^{(\mu+1)}(\nu + 1) + \underbrace{\frac{1}{\Delta t} \cdot \Phi^{(\mu)}(\nu + 1) - \frac{L^{(\mu)}(\nu + 1)}{\Delta t} \cdot i^{(\mu)}(\nu + 1)}_{U_1^{(\mu)}(\nu+1)} - \underbrace{\frac{1}{\Delta t} \cdot \Phi(\nu)}_{U_2(\nu)} \quad (292)$$

In (292), the terms have been ordered in groups of indexes. These groups can be interpreted as the components of an equivalent circuit of the discretized and linearized capacitor as the serial connection of an impedance and two voltage sources. Reformulating (292) with $i^{(\mu+1)}(\nu + 1)$ on the left side results in the following form:

$$i^{(\mu+1)}(\nu + 1) = \underbrace{\frac{\Delta t}{L^{(\mu)}(\nu + 1)}}_{Y^{(\mu)}(\nu+1)} \cdot u^{(\mu+1)}(\nu + 1) \quad (293)$$

$$- \underbrace{\frac{1}{L^{(\mu)}(\nu + 1)} \cdot \Phi^{(\mu)}(\nu + 1) + i^{(\mu)}(\nu + 1)}_{I_1^{(\mu)}(\nu+1)} + \underbrace{\frac{1}{L^{(\mu)}(\nu + 1)} \cdot \Phi(\nu)}_{I_2(\nu)}$$

The abbreviations in (293) refer to the equivalent circuit shown in

Figure 58.

8. Index

accuracy numerical integration.....	72, 76, 80, 83
backward substitution	27, 29
branch admittance matrix.....	11
branch aggregate voltage source vector	16
Branch constitutive equations (BCE)	11
branch current source vector.....	11
branch current vector	10
branch impedance matrix.....	15
branch voltage source vector.....	11
branch voltage vector	10
computational complexity	30
computational order.....	30
controlled sources	18
convergence factor	45
convergence rate.....	45
DC operating point	43
degrees of node.....	37
difference equations.....	63
differential algebraic equation system (DAE)	6
differential branch admittance matrix	55
differential nodal admittance matrix.....	53
directed graph	10
edge set.....	10
energy-storage devices	61
explicit/implicit numerical integration 71, 75, 79, 83	
fill-in	35
Forward Euler	63
Gaussian elimination	22, 27
incidence matrix	12
Jacobian matrix	50
Kirchhoff's current law (KCL).....	11
Kirchhoff's voltage law (KVL).....	11
Laplace transformation.....	61
linear convergence	45
LU decomposition	27, 28
Markovitz product	41
modified nodal analysis (MNA)	16
m-step numerical integration.....	68, 69
natural Pivoting	38
nodal admittance matrix.....	13, 16
nodal current source vector	13, 16
node matrix	11
node voltage vector	10
nonlinear nodal system equations	53
partial fraction decomposition	62, 64
partitioning.....	31
Pivot column.....	29
Pivot row	29
predictor-corrector method	83
quadratic convergence.....	45
root finding.....	43
sensitivity matrix.....	50
setup rules for the modified nodal system	19
stability numerical integration	72, 76, 81, 83
standard branch.....	9
state equations.....	65
structure graph	35
structure matrix.....	35
suboptimal heuristic Pivoting.....	38
subsystem connection	21
superlinear convergence.....	45
transient response	61
vertice set.....	10
Y-branches.....	14
Z-branches.....	14
zero-elements.....	36
z-transformation.....	63