Construction guidelines for a ZPE-Converter on the basis of realistic DFEM-Computations

von Claus W. Turtur Wolfenbüttel, April - 3 - 2011

Ostfalia Hochschule für Angewandte Wissenschaften, Braunschweig-Wolfenbüttel
Salzdahlumer Straße 46 – 48 38302 Wolfenbüttel Germany
Email: c-w.turtur@ostfalia.de Tel.: (+49) 5331 / 939 – 42220
Internet-Seite: http://www.ostfalia.de/cms/de/pws/turtur/FundE

(published at PHILICA.COM, ISSN 1751-3030, Article no. 233)
PACS numbers: 88.05.Ec, 88.05.Gh, 88.05.Np, 88.90.+t

Abstract

In [Tur 11] the theory of a powerful vacuum-energy converter was developed, and such converters have been simulated with a dynamic finite element method (DFEM). The result was a theoretical description of the machine which should be appropriate for technical applications.

Due to many questions from colleagues who read the mentioned article, the author decided to continue his development on the DFEM-algorithm in order to simulate a zero-point-energy (ZPE) motor on the computer, as close to reality as possible.

The theoretical background of the simulation is explained in detail here, so that every colleague should be able, to use the algorithm in the appendix of the publication and to adapted it to the setup of a vacuum-energy motor according to his own conception.

Table of contents

1. Physical fundament and preliminary work

The algorithm is designed to simulate electric and magnetic ZPE-motors by principle, and it is not restricted to one special design or setup. Thus it allows simulating ZPE-motors with arbitrary position and numbers of coils as well as arbitrary positions and numbers of magnets. Even electrostatic ZPE-motors can be simulated. Also interaction with external entities can be included into the simulation, such as connections of coils with electrical circuits.

The functioning principle of the DFEM-algorithm is the following: Any motion (of mechanical components as well as of electrical charge or even of electrical and magnetical fields) is being brought back to differential equations, which also contain the information of external electrical circuits. Permanent magnets have to be simulated by conductor loops containing electrical current

without power supply (which is not an unusual point of view). This allows the algorithm to determine Lorentz-forces, with which external magnetic fields interact with permanent magnets.

In order to present a concrete result (for the suggestion of a prototype), the algorithm which is shown in the appendix, is designed to simulate two coils and one permanent magnet as being drawn in figure 1.

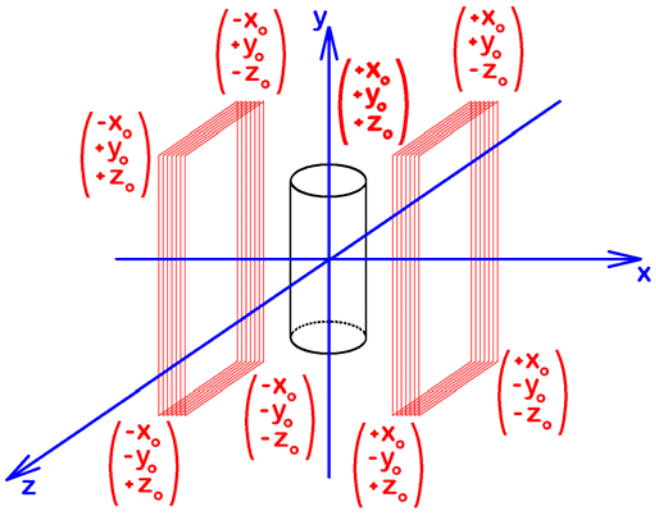


Fig.1:
In a three-dimensional Cartesian coordinate system (blue colour), we see two coils oriented parallel to the yz-plain (red colour). The corners of the coils are located at the Cartesian coordinates as written in red colour. A cylindrical permanent magnet is being simulated by two conductor loops, one at its top end and the other one at its bottom end. The magnet can rotate around the z-axis.
The number and the arrangement of the coils can be chosen arbitrarily in the DFEM-algorithm. The example as being shown here, corresponds to the source-code in the appendix of the publication.

In order to prepare the solution of the differential-equations within the DFEM-algorithm, we need the following:

(a.) The computation of the induced voltage, which the rotating permanent magnet induces in the coils,

and

(b.) The computation of the magnetic force Lorentz-force with which the coils act onto the permanent magnet.

These both computations have the purpose to realise the coupling of the mechanical and the electrical parts of the system to each other (see [Tur 11]). On the one hand, the mechanical rotation of the magnet is influenced by the electrical current in the coils due to the Lorentz-force (between these currents and the permanent magnet), and on the other hand the electrical current in the LC-oscillation-circuit is influenced by the induced voltage which the rotation of the permanent magnet brings into the coils.

We now want to turn our attention to those both calculations as given under (a.) and (b.):

Details of a:

The determination of the voltage induced in the coils (due to the motion of the magnet) is based on the time dependent alteration of the magnetic flux ψ, which has its reason in the rotation of the permanent magnet (in our example around the z-axis), which is normally calculated as being seen in equation (1). [Jac 81]

$$\psi = \int \vec{B} \cdot d\vec{A} \;\Rightarrow\; U_{ind} = -\,\mathrm{N} \cdot \frac{\mathrm{d}\,\psi}{\mathrm{dt}} \qquad (N = \text{number of windings of the coil}) \tag{1}$$

This computation begins with a determination of the magnetic field of the permanent magnet as a vector field which has to be stored in a data-array. This can be done experimentally or theoretically, as for instance with one of the subroutines „Magnetfeld_zuweisen“, which are marked with different numbers in the source-code in order to allow all the emulation of different permanent magnets. The vector field is now fixed rigidly to the permanent magnets so that each of permanent magnets has its own vector field. As soon as we rotate a magnet, the field is rotating together with its magnet. The

rotation around the z-axis is realised with a coordinate-transformation as usual (see equation 2). The angle φ describes the orientation of the length-symmetry-axis of the magnet relatively to the y-axis. (x,y) are the coordinates in the system without rotation (as shown in figure 1), and (x',y') are the coordinates in the system being rotated relatively to (x,y) by an angle of φ. The responsible subroutine in the algorithm has the name „Magnet_drehen".

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(-\varphi) & \sin(-\varphi) \\ -\sin(-\varphi) & \cos(-\varphi) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{Matrix multiplication} \qquad (2)$$

By this means, we calculate the magnetic field strength, which the magnet produces at any arbitrary position in the space, namely as a function of the angle φ. Responsible for the computation of the field strength is the subroutine with the name „Feldstaerke_am_Ort_suchen". Furthermore, at the end of this subroutine, the magnetic flux through the coil is being determined, in which the induced voltage has to be calculated. Therefore we use a subdivision of the coils into finite area-element, so that the magnetic flux can be taken into account as a function of the position, where it passes the coils.

In our very example, this computation is being simplified by the fact, that all orthogonal-vectors on all area elements of each coil are orientated exactly into the direction of the x-axis, so that the scalar-product of the field with orthogonal-vectors of the coil-area-elements can be derived as simple as shown in equation (3), namely as the x-component of the flux.

$$\psi_{SFE} = \vec{B} \cdot d\vec{A} = B_x \cdot \left| d\vec{A} \right| \qquad (3)$$

The magnetic flux through the whole coil is then being calculated as the sum of all magnetic flux-elements through all finite area-elements forming the coil, so that the total flux follows equation (4).

$$\psi_{GES} = \sum \psi_{SFE} \qquad (4)$$

Remark regarding the area-elements of the coil (index „SFE):

In order to formulate the possibilities for the variation of the geometry of the coils as flexible as possible, each coil has to be described as a polygonal line, connecting arbitrarily defined support points. Each coil is modulated connecting its support points with each other. This allows the definition of arbitrarily shaped areas to be surrounded by coils. Finite conductor-loop elements are defined by the geometrical connections between support point and support point. Finite area-elements of the coils fill up the area surrounded by the conductor-loops, so that the magnetic flux through each coil can be calculated as the sum of the finite magnetic flux elements through all area-elements of the individual coil.

From this result as described in equation (4), the calculation of the induced voltage is simply a derivation to time as shown in equation (1). Therefore we need the time-dependency of the orientation of the magnet (given by the angle φ), this is the angular velocity of the permanent magnet at each moment of time. If we check our calculation with a constant angular velocity of the magnet, we come to a result as shown in figure (2).

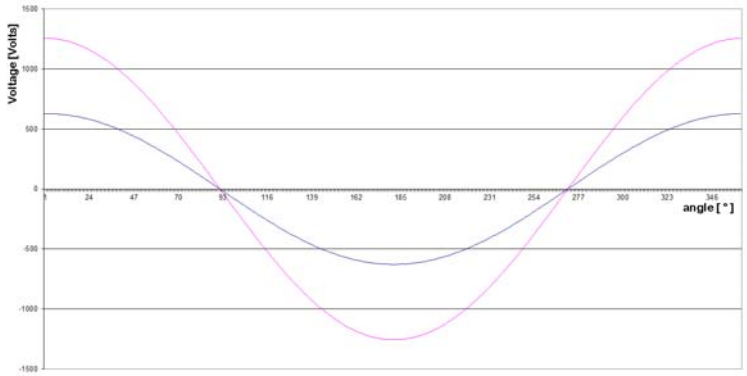


Fig.2:
For the verification of the computation-method, a permanent magnet producing a homogeneous magnetic field was rotated with constant angular velocity, and the induced voltage in both red coils of figure 1 was plotted.

The fact, that both voltages differ by a factor of 2 (from coil to coil) has its reason in the fact, that one coil has twice as many windings as the other one.

For additional computations (see b.), we need to determine the torque which the magnet experiences due to the electric current in the coils. This requirement makes it necessary to emulate the permanent magnet by a configuration of conductor loops. If the magnetic field is not just a simple homogeneous field (on which fig.2 is based), the computation of the magnetic flux depends on the spatial resolution of the computation of the magnetic field. Due to this reason, the magnetic flux of more complicated magnets always display some numerical noise (due to the fact, that the finite elements are not continuous but disctere in spatial resolution). And the problem is, that the numerical noise is enhanced remarkably, when we calculate the derivative to time. An example for such numerical noise is shown in figure 3, as it was calculated by the subroutine „Magnetfeld_zuweisen_02“. When we will emulate a real cylindrical bar-magnet later (with the subroutine “Magnetfeld_zuweisen_03”), the numerical noise will be even much worse.

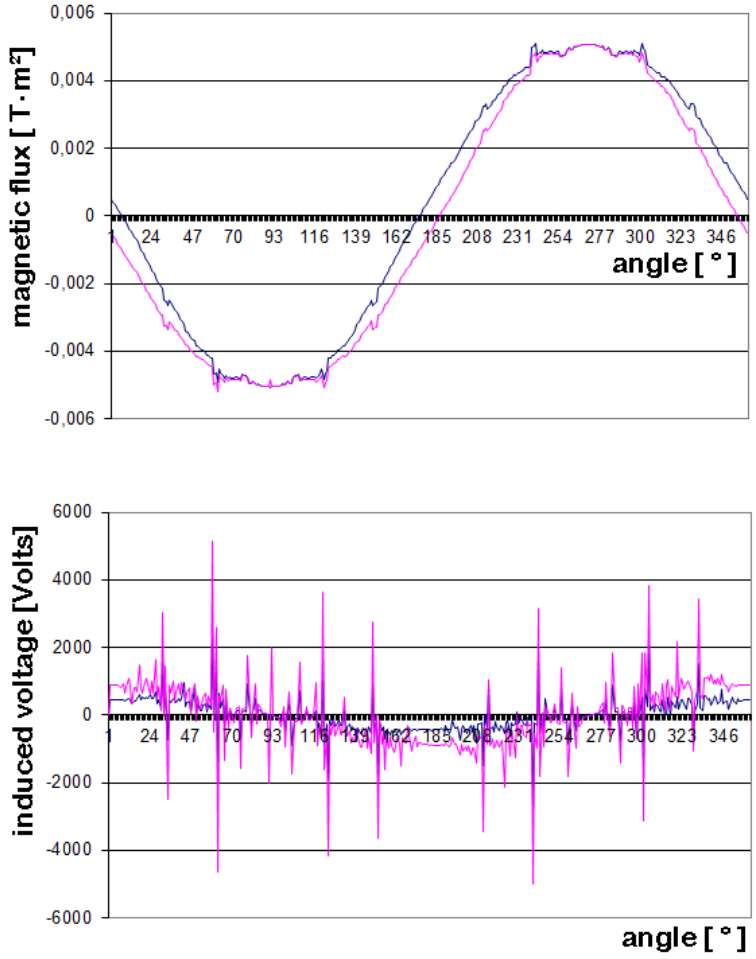


Fig.3a:
Although the magnetic flux from a magnet rotating with constant angular-velocity has only moderate numerical noise, the computation of the induced voltage needs numerical smoothing urgently. Otherwise it would not be sufficient for any use in the DFEM-algorithm.

In order to smooth the numerical noise of the voltage-signal, a Fourier-series was developed (in the subroutine “Fourier_Entwicklung”). It is important to take only low order components (maximum up to fifth order), in order to assure, that high frequency components are excluded. The less high-order components we take, the smoother the signal we get.

An additional effect of the Fourier-series is, that it helps to save CPU-time remarkably, when we will have to calculate the magnetic flux very often for the solution of the differential equation later. The computation of the magnetic flux itself contains a sum (see equation 4), which contains time-consuming operations (see equations 2 and 3), which take much more CPU-time, then the calculation of the Fourier-series with only five rather simple expressions to be summed up. This is important, because the solution of the differential-equation will have to be done with very many time-steps of few nanoseconds, so that the computation of the magnetic flux has to be done several 10^8 or 10^9

during each run of the DFEM-algorithm. This forces us to speed up the very innerst loops in the program, as the computation of the magnetic flux is one of them.

$$\psi_{GES} = \sum_{\nu=1}^{N_0} A_\nu \sin(\nu\,\omega t) \qquad \begin{matrix}\text{approximation by Fourier-series in 5.order}\\ A_\nu = \text{Fourier-coefficients}\end{matrix} \qquad \text{(see [Bro 08])} \qquad (5)$$

The fact that the Fourier-series approximation is done in rather low order ($N_0 \le 5$) allows us to determine the Fourier-coefficients very easy by the use of the Gauß'ian method of the least square fit, to compare the Fourier approximation with the original data. This method is used to determine the Fourier-coefficients A_ν .

Details of b:

For the determination of the Lorentz-forces, by which the electrical currents in the coils accelerate the permanent magnets, the emulation of the permanent magnets have being realised (as mentioned above) by conductor loops containing electrical current. In our example for the demonstration, we want to emulate cylindrical bar-magnets, because they are easy to buy and not very expensive, with regard to experiments. For the sake of simplicity, the cylindrical bar-magnets are emulated by two circular conductor loops, located at each end of the cylindrical bar. This means, the magnetic field of the bar-magnets is emulated by the magnetic field of one pair of circular coils.

The parameters we need are only the length of the bar, the diameter of the bar and the magnetic field at each end of the bar. Especially the magnetic field-strength can be adjusted by the choice of the electric current in the conductor loops emulating the bar-magnet.

The calculation of the magnetic field of the emulation-conductor-loops is done with the use of Biot-Savart's law. The approach is illustrated in figure 4.

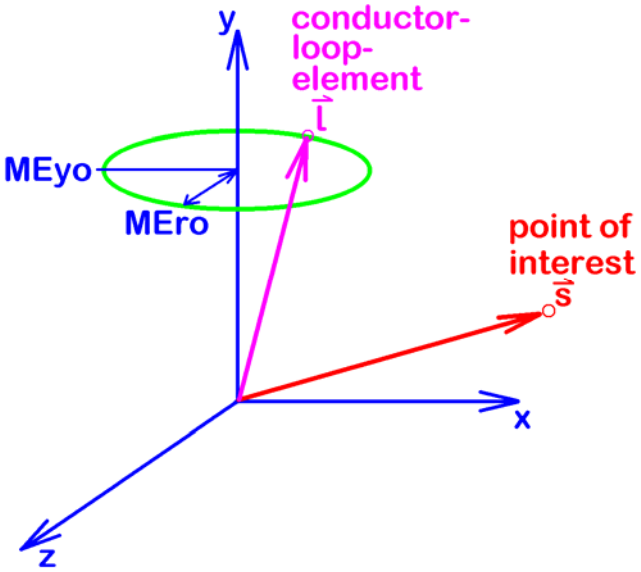


Fig.4:
Illustration of the geometry of a conductor loop (green) whose elements are parametrized by a position vector $\vec{l}$. According to Biot-Savart, we calculate the magnetic field, which the conductor loop produces at any arbitrary point of interest $\vec{s}$.
The conductor loop shown here describes the top end of the cylindrical bar-magnet, which is orientated along the y-axis. Thus the loop is orientated parallel to the xz-plane, has the radius „MEro", and is located at the y-position „MEyo".

The parameterisation of the conductor loop can be realised rather simple according to equation 6.

$$\vec{l}(t) = \begin{pmatrix} MEro\cdot\cos(\omega t+\varphi) \\ MEyo \\ MEro\cdot\sin(\omega t+\varphi) \end{pmatrix} \quad \Rightarrow \quad \vec{v}(t) = \frac{d}{dt}\vec{l}(t) = \begin{pmatrix} -\omega\cdot MEro\cdot\sin(\omega t+\varphi) \\ 0 \\ +\omega\cdot MEro\cdot\cos(\omega t+\varphi) \end{pmatrix} \qquad (6)$$

Giving the point of interest $\vec{s} = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$ in cartesian coordinates, we can introduce equation (6) into the law of Biot-Savart:

$$d\vec{H} = \frac{q_1 \cdot \vec{v} \times \left(\vec{l} - \vec{s}\right)}{4\pi \cdot \left|\vec{l} - \vec{s}\right|^3} \cdot \frac{d\varphi}{2\pi} \tag{7}$$

The outer-product in the counter of equation (7) is

$$\vec{v} \times \left(\vec{l} - \vec{s}\right) = \begin{pmatrix} -\omega \cdot MEro \cdot \cos(\omega t + \varphi) \cdot \left[MEyo - s_y\right] \\ \omega \cdot MEro \cdot \cos(\omega t + \varphi) \cdot \left[MEro \cdot \cos(\omega t + \varphi) - s_x\right] + \omega \cdot MEro \cdot \sin(\omega t + \varphi) \cdot \left[MEro \cdot \sin(\omega t + \varphi) - s_z\right] \\ -\omega \cdot MEro \cdot \sin(\omega t + \varphi) \cdot \left[MEyo - s_y\right] \end{pmatrix} \tag{8}$$

The absolute value of the denominator in equation (7) is

$$\left|\vec{l} - \vec{s}\right|^3 = \left(\left[MEro \cdot \cos(\omega t + \varphi) - s_x\right]^2 + \left[MEyo - s_y\right]^2 + \left[MEro \cdot \sin(\omega t + \varphi) - s_z\right]^2 \right)^{3/2} \tag{9}$$

In principle, we now can introduce the expressions of (8) and (9) into the outer product of (7), but in order to make (6) complete, we additionally need the electrical charge q_1 in (8). This has to be determined from the current I in the coil and the propagation-velocity of the electrical charge, as being described by the angular velocity ω. Therefore I and ω have to be combined in such a way, that the motion of the electrical charge is being described appropriately. As we know, the electrical current is defined as the amount of electrical charge flowing per time. Thus we can write:

$$\left.\begin{aligned} I &= \frac{q_1}{T} \\ \mathrm{T} &= \frac{2\pi}{\omega} \end{aligned}\right\} \quad \Rightarrow \quad I = \frac{\omega}{2\pi} \cdot q_1 \tag{10}$$

Therefore either q_1 or ω can be chosen arbitrarily, and the other one has to be adjusted adequately, so that the electrical current I is correct to produce of the magnetic field which has to be emulated. We decide to chose arbitrarily $q_1 = 1\ Ampere$, and to adjust ω. This can be done by a calculating ω from equation 10 as being shown in equation (11):

$$\omega = \frac{2\pi \cdot I}{q_1} \tag{11}$$

For the summation of the infinitesimal field-elements of equation (7), we could in principle solve the integral of equation (12). But the algorithm is designed for arbitrarily shaped conductor loops, and we already have N discrete finite elements, so that we can solve equation (12) by an approximation of a discrete sum as also shown in the same equation (12). But we shall keep in mind that a discrete sum always makes numerical noise (similar as shown in figure 3).

$$\vec{H}_{GES} = \underset{\substack{Leiter-\\schleife}}{\oint} d\vec{H} \approx \sum_{i=0}^{N} d\vec{H} = \sum_{i=0}^{N} \frac{q_1 \cdot \vec{v} \times \left(\vec{l} - \vec{s}\right)}{4\pi \cdot \left|\vec{l} - \vec{s}\right|^3} \cdot \frac{\Delta\varphi_i}{2\pi} \tag{12}$$

The summation has been realised within the subroutine „Magnetfeld_zuweisen_03“, with the variable of summation being $\mathrm{I} = 0 \ldots N$, with the aim to make the argument of the parametrization run from $\mathrm{t} = 0 \ldots T = \frac{2\pi}{\omega}$ (for detailed understanding, please see subroutine in the appendix).

The results have been checked by the classical formula for the calculation of the field-strength (see equation 13) along the axis of the coil (which here is the y-axis), and the check confirms our results.

$$\vec{H}_{Klass} \approx \frac{I \cdot a^2}{2 \cdot \left(a^2 + r^2\right)^{3/2}} \tag{13}$$

Now our simulation of the magnetic field of a cylindrical bar-magnet by the use of two conductor loops at each end of the cylinder is complete.

The determination of the magnetic field of a permanent magnet is not our goal, but it is one important step on our way towards the goal. Our goal finally is the determination of the torque, with which the coil accelerates or decelerates the rotating permanent magnet. Therefore we have to determine the Lorentz-force with which the currents in the coils (red colour in figure 1) does act onto the conductor loops emulating the permanent magnet. With other words: We have to calculate the Lorentz-force, which is the fundament for the calculation of the torque, which the permanent magnet experiences.

Therefore we again use Biot-Savart's law. Now we apply it in a way that we calculate the magnetic field produced by the red coils at the position of the permanent magnet emulation conductor loops, which have been used to emulate the permanent magnet. This is necessary that we can calculate the Lorentz-force, which the permanent magnet emulation conductor loops experience within this field of the red coils

This means, that the conductor loops producing the field to be calculated now, are described by the polynomial line of the red coil, and the points of interest at which the field has to be calculated is the position of each conductor loop-element, which experiences a Lorentz-force. The situation is illustrated in figure 5.

Due to the rectangular shape of the red loops, we could in principle use a polynomial line with only for support points. But in reality this is not sensible, because we need to have several (many) finite area-elements within the red coils, so that the magnetic flux through the coils will be calculated properly (see equation 4).

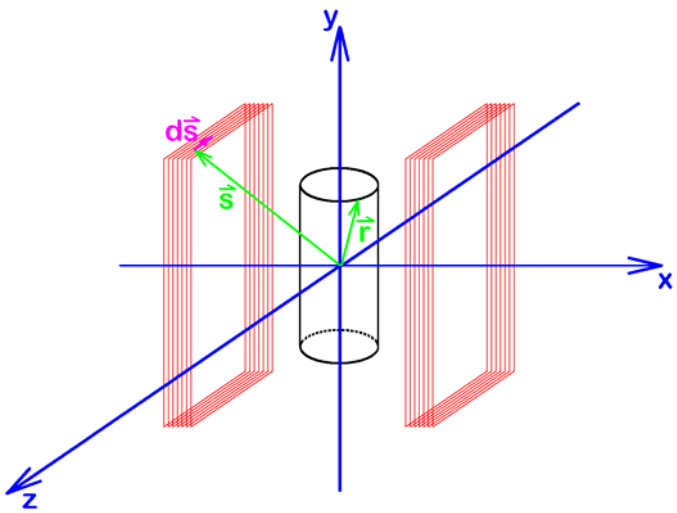


Fig.5:
Illustration of the vectors as being used for the calculation of the magnetic fields of the red coils at the position of the conductor loops emulating the permanent magnet as being used for the application of Biot-Savart's law.

$\vec{s} = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$ conductor-loop element

$\vec{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}$ point of interest

$d\vec{s} = \begin{pmatrix} d\vec{s}_x \\ d\vec{s}_y \\ d\vec{s}_z \end{pmatrix}$ motion of the electrical charges in the conductor loop-element

With regard to the parameters according to figure 5, we can write Biot-Savart's law according to equation (14):

$$d\vec{H} = \frac{I \cdot d\vec{s} \times \left(\vec{s} - \vec{r}\right)}{4\pi \cdot \left|\vec{s} - \vec{r}\right|^3} \tag{14}$$

The outer product in the counter is

$$d\vec{s} \times \left(\vec{s} - \vec{r}\right) = \begin{pmatrix} d\vec{s}_x \\ d\vec{s}_y \\ d\vec{s}_z \end{pmatrix} \times \begin{pmatrix} s_x - r_x \\ s_y - r_y \\ s_z - r_z \end{pmatrix} = \begin{pmatrix} d\vec{s}_y \cdot \left(s_z - r_z\right) - d\vec{s}_z \cdot \left(s_y - r_y\right) \\ d\vec{s}_z \cdot \left(s_x - r_x\right) - d\vec{s}_x \cdot \left(s_z - r_z\right) \\ d\vec{s}_x \cdot \left(s_y - r_y\right) - d\vec{s}_y \cdot \left(s_x - r_x\right) \end{pmatrix} \tag{15}$$

As usual, we put the expression of (15) into equation (14) to calculate the finite field-elements $d\vec{H}$, which are produced by all finite conductor elements of the red coils. This is the way how we come to the field which the red coils produce at the position of the magnet-emulation-coils. This calculation has to be done individually for each loop-elements of the magnet-emulation-coils. From there we come to Lorentz-force acting onto each magnet-emulation-coil (see equation 16).

$$d\vec{F} = q \cdot \left(\vec{v} \times d\vec{B}\right) = I \cdot \left(\vec{l} \times d\vec{B}\right)$$
$$\text{with the field elements } d\vec{B} = \mu \cdot d\vec{H} \tag{16}$$

From all finite Lorentz-force-elements $d\vec{F}$ we now calculate the finite torque-elements with which they act onto the rotation of the permanent magnet. The summation of these torque elements (as a discrete sum, see equation (17)) delivers the torque on the rotating magnet as being used in the DFEM-algorithm.

$$d\vec{M} = \vec{r} \times d\vec{F} \qquad \left(\text{finite torque-element}\right)$$
$$\vec{M}_{ges} = \sum d\vec{M} \qquad \left(\text{Summation for the total torque, approximation by discrete sum}\right) \tag{17}$$

The calculations are realised in the subroutine „Drehmoment“.

Due to the spatial discretization, we also have numerical noise which has to be smoothed by a Fourier-series. Therefore we develop the torque as a function of the angle φ of the orientation of the magnet. We again restrict ourselves to fifth order or less in order to exclude high-frequency components for sure (compare (5)).

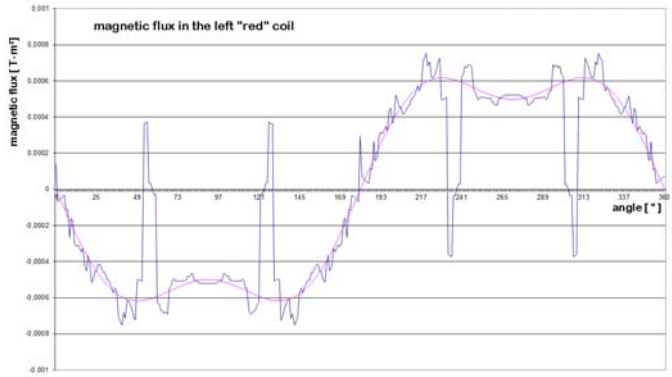
Again, the elapsed CPU-time to calculate the torque is rather large, so that the explicit torque-calculation is not recommendable within the solution of the differential-equation. Here we have again the advantage to save computer time due to the Fourier-series.

Now we reach the point, that the preparations are complete, as they are the following two steps:

- the calculation of the induced voltage, which the rotation of the permanent magnet induces into the coils, and
- the computation of the torque, with which the electrical current in the coils act onto the permanent magnet.

The subroutines which do this calculations in fast manner (due to the quick Fourier-series) have the names „Schnell_Drehmoment“, „Fluss_T“ and „Fluss_I“. They can be seen in the source code in the appendix.

We summarise the results of section 1 in figure 6 and figure 7, which corresponds to the geometry of figure 5.



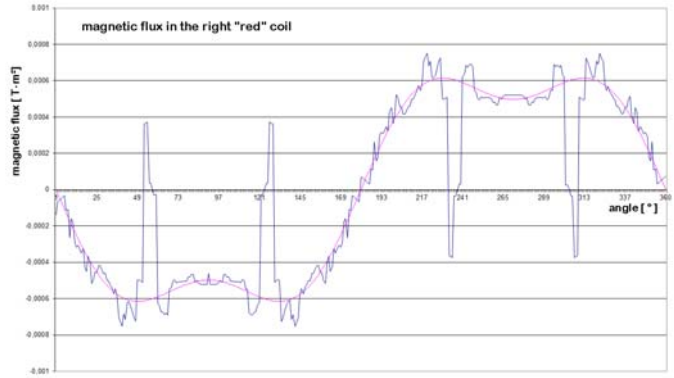


Fig.6:
The magnetic flux, which the permanent magnet according to figure 5 produces in the red coils depends on the orientation of the permanent magnet (angle).
Due to the symmetry of the setup, the magnetic flux has the same value for both coils.
The blue signal allows us to estimate the numerical noise, which has its reason in the spatial discretization of the geometry. The maximum of the numerical noise occurs in the moment, when the windings of the magnet emulation coils, come most close to the wires of the red coils.
The purple signal is a smoothing of the blue signal as being calculated by the approximation of a 5th order Fourier series.

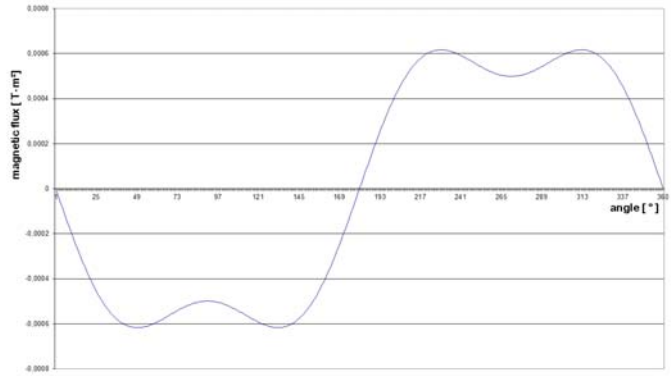
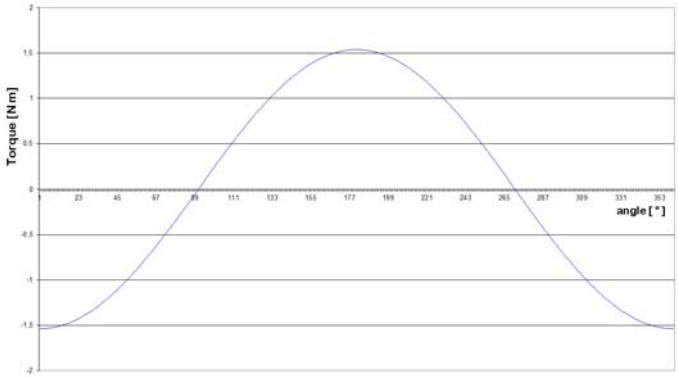


Fig.7:

If we later derive the magnetic flux to time, we calculate the induced voltage in the red coils (later, not printed here). Therefore we need to know the angular velocity of the rotating magnet.

The curve of the torque is made to check the computation. Therefore the red coils are driven with a constant current, so that the torque depends mainly on the orientation of the magnet.



The values for the parameters are:
- permanent magnet, cylindrically, 4 cm thick, 8 cm long, field strength 1 Tesla at ist end.
- „Red“ coils, rectengular, 6 cm wide, 12 cm height, located at x=–2 cm and at x=+2cm.

We explicitely emphasize, that the number of coils can be chosen arbitrarily same as the number of permanent magnets within the DFEM-algorithm presented here. This means that the algorithm can be used for the computation of every ZPE-converter working electrically or magnetically:

- If the DFEM-algorithm is used to emulate several permanent magnets, their interaction can be analysed. This is an interesting application of the algorithm for the simulation of self-running ZPE-magnetmotors, as they can for instance be found at [Hoh 11], [Jeb 06].
- If the DFEM-algorithm is used to emulate several coils, which might be connected with each other by a yoke if requested (in order to conduct the magnetic flux in an appropriate way), motionless ZPE-converters can be simulated, for instance as can be seen at [Mar 88-98], [Bea 02].
- If the DFEM-algorithm is used to emulate one coil and one permanent magnet, the "Electro-Mechanic Double Resonance" converter (EMDR-converter) as proposed by the author of this article can be simulated [Tur 11].
- If the DFEM-algorithm is used to emulate two coils and several permanent magnets fixed to each other in appropriate manner, the Keppe-motor can be simulated [Kep 10].
- If 6 cylindrical bar-magnets are mounted within 6 coils and are arranged with each other in a hexagon, the Coler-apparatus can be simulated. The behaviour of these elements will lead us to a rather complicated differential-equation, and some electrical elements forming an electric circuit are introduced as boundary conditions into the system of differential-equations. (For differential-equations, please see also section 2.) Perhaps, a simulation of the Coler-apparatus might help to decide whether this motionless-converter can work or not. [Hur 40], [Mie 84], [Nie 83]
- Also dynamic input and output of energy is no problem, because the differential-equations describing the motion can be expanded with some input- voltages, load-resistors, an so on… Such elements have to be taken into the differential-equations additionally. Also mechanical force of torque can be applied as boundary conditions in the differential-equations. In order to illustrate this, the source-code in the appendix contains two coils (see fig.5). In the source code, the left coil has of the name input-coil on the right coil has the name turbo-coil (see subroutine "U7"). Additionally there is a load-resistor („R_{Last}") being connected with the turbo-coil. The rotating axis of the permanent magnet is being supported initially with a given angular velocity, being applied as initial-condition for the solution of the differential-equation. This initial rotation brings energy into the system once at the very beginning of the motion and can then be disconnected. This means that the EMDR-converter is a self running motor, which needs energy support only at the very beginning of the motion to initialise the rotation. Furthermore the input-coil is not active in the source code as printed in the appendix, because of EMDR-converter does not need permanent input-energy, for it is a self running engine. Nevertheless the input-coil can be used if somebody wants to do this, as for instance for the purpose to control the "rounds per minute" of the motor.
- Furthermore the algorithm allows mechanical extraction of power. In our differential-equation this is simulated by a decelerating torque proportional to the angular velocity of the rotation (of the magnet). Such type of energy-extraction can be used to simulate friction as well as to simulate energy-extraction for technical application.
- In order to develop an exemplary ZPE-converter and bring it into a stable permanent mode of operation, it is a convenient method, to control some input-power or to control the extraction of power. With regard to a self running engine, there is no input power, so that we decided to control the output-power in the DFEM-algorithm shown in the appendix. Therefore we define a special value for the speed of revolution (rounds per minute). If the rotation is faster, the energy extraction is enhanced, and of the rotation is less fast, the energy extraction has to be reduced (respecting a given hysteresis, necessary for proper switching). Later we will find out, that our exemplary ZPE-motor can also work without a regulation, but with constand extraction of energy.

The variability of the DFEM-method is large, so that it is not restricted only to magnetic ZPE-converters, but it is also applicable to electrostatic ZPE-converters. The only necessary change is, to replace the Lorentz-force from equation (16) by the Coulomb-force as seen in equation (18).

$$d\vec{F}_{12} = \frac{Q_1 \cdot dQ_2}{4\pi\varepsilon_0} \cdot \frac{\vec{r}_1 - \vec{r}_2}{\left|\vec{r}_1 - \vec{r}_2\right|^3} \qquad \begin{array}{l}\text{with finite force-elements,}\\ \text{by which the charge } Q_1 \text{ acts}\\ \text{on finite charge elements } dQ_2.\end{array} \tag{18}$$

The precision of the calculation mainly depends on the precision of the input-data, namely the data of the mechanical and electrical components as well as of the data of the interacting fields with which the components act onto each other.

2. Motion of the components of the ZPE-Converter

In section 1 we did the preparation of the necessary fundamental equations of physics. This is now done, and we can turn our attention towards the solution of the differential-equations describing the motion in the ZPE-converter. Hereby we speak about motions of the mechanical components as well as about motions of the electrical components (such as electrical charges and fields or magnetic fields).

The functioning principle of every ZPE-converter can be described by the motion of its components. The adequate means for this description are the differential-equations based on the interactions of the components with each other.

For ZPE-converters of course do not consist only of one single component, but of several components, which have to interact with each other, we always have two put up and solve coupled systems of differential-equations of higher order. If converters need input-energy (such types which do not work as self running engines, but only as over-unity systems), the energy-input has to be introduced as perturbation-function in the differential-equations. For all types of interaction with some external elements components, the appropriate means is the introduction of perturbation-functions into the differential-equations. This makes the higher order differential-equation systems inhomogeneous.

Mathematically, this has the consequence, that we cannot simply derive an analytical solution, valid for each type of differential-equation system. Consequently, the central core of computation of the DFEM-algorithm is a numerical iterative solver of the differential equations. This topic is, to what we want to focus our attention in section 2.

On this background it is clear, that the DFEM-algorithm needs some certain amount of CPU-time, if you want to have the solution with sufficient precision. You cannot expect the DFEM-algorithm to come to good convergence within few seconds.

The understanding, how the differential-equations of the motions have to be formulated, is the central point of understanding, which every user of the DFEM-algorithm has to work on. Only on the basis of this understanding, the user can apply the DFEM-algorithm onto his own system.

Let us now begin to develop the differential-equations for the example of the EMDR-converter, which define the core of computation of the algorithm. We will do this in analogy to [Tur 11].

The differential-equations by principle operate completely dynamically, so that the DFEM-algorithm has to do the same (taking the finite speed of propagation of the interacting fields into account). This has the consequence, that all physical sizes, entities and values under consideration have to be drawn back to the oscillating electrical charges or to the rotating magnet, using the fundamental

entities of q, $\frac{d}{dt}q = \dot{q}$, $\frac{d^2}{dt^2}q = \ddot{q}$ and of φ, $\frac{d}{dt}\varphi = \dot{\varphi}$, $\frac{d^2}{dt^2}\varphi = \ddot{\varphi}$, where q if the electrical charge and φ is the angle of rotation of the permanent magnet.

From [Tur 11] we learned, that it is necessary to have very fine time-steps in order to get reliable results. Consequently it will not be possible to save all data of all time-steps within special data-arrays. So the program now allows the calculation of as many time steps as required, but the data-storage will be done only for maximum 35,000 points which is a sensible upper limit for the data-export to Excel.

Values as for instance the inductivity of the cylindrical coil (see equation 19) or the momentum of inertia of the rotating magnet as a massive cylinder (see equation 20) are taken from standard textbooks of physics or engineering disciplines.

Inductivity $L = \mu \cdot \frac{N^2\ A}{s}$, with N = number of windings

A = cross-section area of the coil [Ger 95] (19 a)

s = length of the coil-body

Or more precise for short coils:

Inductivity $L = \mu \cdot \frac{N^2\ A}{\sqrt{s^2 + \frac{A}{\pi/4}}}$, *mit* N = number of windings (19 b)

A = cross-section area of the coil to be derived from [Stö 07]

s = length of the coil-body

$J_y = \frac{m}{4} \cdot \left(r_a^2 + r_i^2 + \frac{h^2}{3} \right)$ moment of inertia of rotation of a massive cylinder rotating around a axis perpendicular to its length [Dub 90] (20)

On this basis we can simulate a setup according to figure 8.

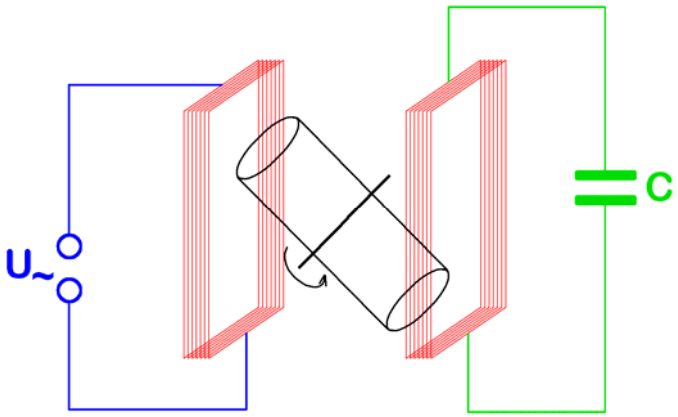


Fig.8:
This is the setup of a ZPE-converter for whose simulation we want to develop the DFEM differential-equations.
It consists of two coils (red colour) and one rotating permanent magnet (black colour) and additionally an input-voltage (blue colour) and one capacitor (green colour).

For our EMDR-converter as our very example of a ZPE-converter can be operated as a self running motor, the input-voltage is not applied at all, so that the left coil together with voltage-supply is taken away completely. Within the source-code, those lines for the simulation of the input-voltage-components are left away; they are marked as comment so that they do not take part at the calculation at all. In order to make the setup symmetrically, the right coil was moved symmetrically around the axis of rotation of the permanent magnet. In order to make clear to everybody how the setup is now looking, figure 9 was drawn. Those elements which are not necessary for the simulation of our exemplary EMDR-converter are only printed as comments in the source-code of the algorithm for those who want to use the algorithm for other ZPE-motors. These comments have only the purpose to help colleagues to simulate their ZPE-machines.

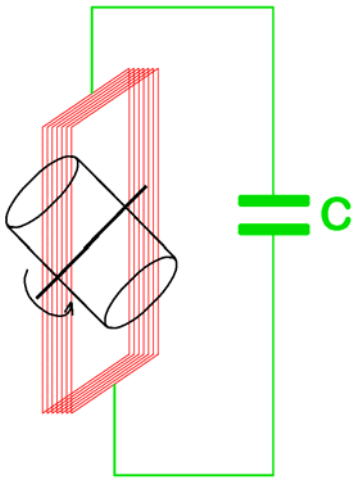


Fig.9:
This very simple setup of an EMDR-converter already allows the powerful conversion of ZPE-energy into classical electrical and classical mechanical energy. Its differential equation system was formulated and solved as explained on the following pages, together with the results describing the behaviour of this ZPE-motor.

The development and formulation of the differential equation system is done rather similar as in [Tur 11].

(a.)

The differential equation of a harmonic oscillation of an electric LC-oscillation circuit is described by equation (21).

$$-L\cdot\ddot{Q}+\frac{1}{C}\cdot Q=0 \;\Rightarrow\; \ddot{Q}=\frac{1}{LC}\cdot Q \qquad (21)$$

(b.)

The differential equation of an attenuated oscillation of an electric LCR-circuit is described by equation (22).

$$\ddot{Q}=-\frac{1}{LC}\cdot Q+\frac{R}{L}\cdot\dot{Q} \qquad (22)$$

Its numerical iterative solution, as being achieved with the solver of the differential equations in our DFEM-algorithm, is described in equation (23) in analogy with [Tur 11], which is in good agreement with the classical solution (see figure 10).

$$Q_i=\dot{Q}_{i-1}+\ddot{Q}_i\cdot\Delta t-\frac{R}{L}\cdot\dot{Q}_{i-1}\cdot\Delta t \qquad (23)$$

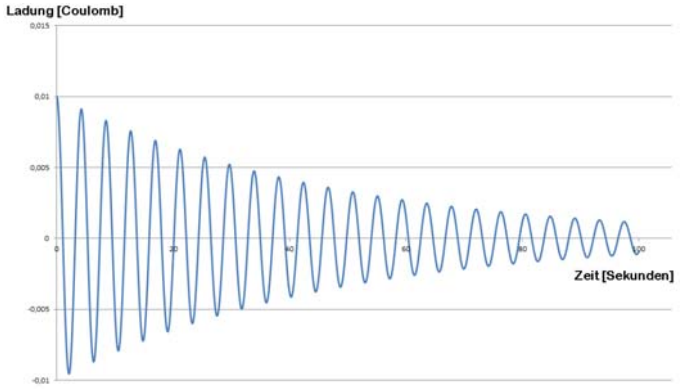


Fig.10:
Verification of an attenuated oscillation in the electric LCR-oscillation circuit for the check of the differential equation (22) and the solution (23).

(c.)

Different from the electrical oscillation, the mechanical motion (of the permanent magnet) is not an oscillation but a rotation and thus it has no restituive force. So we have to take two contributions into account to describe the torque $\vec{M}$. The first contribution goes back to the magnetic field from the coil, which acts onto the permanent magnet. The second contribution goes back to the mechanical extraction of power, which will be the dominant part of the output-power of our ZPE-motor. This second contribution is made by the torque proportional to the angular velocity of the rotation. (Its mechanism could be friction or some other mechanism as well.) The first mentioned contribution is well-known from section 1. The last mentioned contribution will be discussed in detail in section 6.

Consequently the differential equation of the mechanical part of the system can be integrated rather simply as shown in equation (24). This also indicates that equation (24a) contains the coupling of the electrical part of the system into the mechanical part of the system.

$$\ddot{\varphi}(t) = \frac{M(t)}{J} \qquad with\ \ M(t) = torque \quad and \quad J = moment\ of\ inertia \tag{24a}$$

$$\Rightarrow \quad \dot{\varphi}(t) = \int_0^t \ddot{\varphi}(\tau)\, d\tau \quad \Rightarrow \quad numerical\ \ Iteration \quad \dot{\varphi}(t) = \underbrace{\ddot{\varphi}(t)\cdot dt}_{\text{Integration}} + \underbrace{\dot{\varphi}(t - dt)}_{\text{Integration-constant}} \tag{24b}$$

$$\Rightarrow \quad \varphi(t) = \int_0^t \dot{\varphi}(\tau)\, d\tau \quad \Rightarrow \quad numerical\ \ Iteration \quad \varphi(t) = \underbrace{\dot{\varphi}(t)\cdot dt}_{\text{Integration}} + \underbrace{\varphi(t - dt)}_{\text{Integration-constant}} \tag{24c}$$

(d.)

Still missing in our system of differential equations, and thus to be introduced now, is the coupling of the mechanical part of the system into the electrical part of the system. The appropriate means therefore is the induced voltage (and this is the reason, why we did its calculation), which the rotation of the permanent magnet brings into the coil. Therefore the differential equation of the electrical system is expanded, from equation (22), so that we come to equation (25).

$$\ddot{Q} = -\frac{1}{LC}\cdot Q + \frac{R}{L}\cdot \dot{Q} - \frac{U_{ind}}{L} \tag{25}$$

(e.)

On this basis, the numerical-iterative solution of the electrical part of the differential equation system can be integrated according to equation (26) - and by the way in analogy to [Tur 11].

$$\underbrace{\ddot{Q}(t) = -\frac{1}{LC}\cdot Q}_{\text{coil and capacitor}} + \underbrace{\frac{R}{L}\cdot \dot{Q}}_{\substack{\text{Ohm´ian}\\ \text{resistor}}} - \underbrace{\frac{U_{ind}}{L}}_{\substack{\text{Induced}\\ \text{voltage}}} \tag{26a}$$

$$\Rightarrow \quad \dot{Q}(t) = \int_{t-\Delta t}^{t} \ddot{Q}(\tau)\, d\tau \quad \Rightarrow \quad numerical\ \ Iteration \quad \dot{Q}(t) = \underbrace{\ddot{Q}(t)\cdot dt}_{\text{Integration}} - \underbrace{\dot{Q}(t - dt)}_{\text{Integration-constant}} \tag{26b}$$

$$\Rightarrow \quad Q(t) = \int_{t-\Delta t}^{t} \dot{Q}(\tau)\, d\tau \quad \Rightarrow \quad numerical\ \ Iteration \quad Q(t) = \underbrace{\dot{Q}(t)\cdot dt}_{\text{Integration}} + \underbrace{Q(t - dt)}_{\text{Integration-constant}} \tag{26c}$$

This was the explanation, how the main program of the DFEM-algorithm performs the calculation of the motions of the components of the ZPE-converter, analysing on the one hand the motion of the electrical charges and on the other hand the motion of the rotating magnet. This is a dynamical computation. Therefore the algorithm has the dynamical FEM (=DFEM).

General remark:

Within our calculations it is only allowed to use formulas which follow the dynamics of the system. Also the physical sizes used in these formulas have to respect that this criterion. This means that physical sizes as for instance the mean value of an electrical current, some effective values, and so on… are strictly forbidden. Even physical sizes, entities and values, which somehow refer to a special signal-shape are not allowed, because they do not follow the full dynamics of the differential equations. Those who want to adopt the DFEM-algorithm to their own machines and/or experiments have to be very careful, to respect this criterion without any exception. This is important.

Fundamental philosophical remark regarding the propagation of the fields (and first of all the speed of propagation of these fields) within the electrical circuit:

An electrical LC-oscillation-circuit (which for instance can be seen in figure 9) has an electrical oscillation within the coil and the capacitor.

The question is now: Which type of physical entities do oscillate in this circuit ?

Is it electrical charges which oscillate back and forth ?

No - this is for sure not the case ! This can be understood rather easy, when we follow the electrical charges within the LC-circuit, beginning at the moment of time, at which the coil is free from any electrical current. This is the moment, in which the capacitor is charged up to its maximum voltage and charge, so that the total energy of the oscillation circuit is now stored as electrostatic field energy of the electrostatic field between the capacitor plates. From now on, the capacitor begins to discharge, so that electrical fields (and voltages) propagate along the wire of the coil. From the positive capacitor plate one field propagates towards the negative capacitor plate, and in the opposite way another field propagates from the negative capacitor plate into the direction towards the positive capacitor plate. But the propagating entities are only fields, not charge-carriers (such as for instance electrons or electron holes). There are two reasons explaining this argument: The first reason is, that the charge-carriers can not propagate with the speed of $v = 1/\sqrt{LC}$, which we know to be the speed of the propagating electrical signal. The second reason is, that positive and negative charge-carrier would compensate each other as soon as they meet each other in the middle of the coil. If this would happen (as for instance electrons and electron holes would compensate each other), the complete oscillation would stop as soon as the charge carriers from the different capacitor plates meet each other. This would be the case after one quarter of a period of oscillation. Everybody knows that this is in contradiction with the real observation (at LC-oscillation circuits), so obviously the oscillating physical entities are not charge carriers.

But which are the oscillating physical entities - is it electrical fields ?

Yes - this is indeed the case ! Among the typical properties of electrical fields (as well as electrical waves) is the ability to superpose without disturbing each other, and even to cross the path of each other without taking notice of each other. So we can imagine one field as a wave crest and the other one as a way through, both of them moving from one capacitor plate to the opposite plate, not interfering with each other when they pass the coil at the same moment. Both of them follow their speed of propagation $v = 1/\sqrt{LC}$, and thus the oscillation circuit behaves as we know it from wave theory.

Because of their ability to pass each other without disturbing each other, the positive wave crest and the negative way through reaches the opposite capacitor-plate after half an oscillation, and they take the other half of the oscillation to find their way back "home", and so on… (cyclically…).

Of course the propagating fields cause small displacements of the charge-carriers within the wire of the coil, but these displacements are rather small - by several orders of magnitude too small to transport charge-from one capacitor plate the opposite one.

We can understand this rather easy, when we have a look to the acoustic analogon. If an acoustic signal propagates within a tube (the one-dimensional consideration makes it easier, and it is in good agreement with the one-dimensional propagation of the electrical signals in the wire), we can also send the wave crest and the wave trough from the opposite ends of the tube, and we will also see, that they pass each other without disturbing each other. What we regard here is the field of air-pressure, in which waves always superpose without disturbing each other. The conception is the following: The tube can be subdivided in small finite volumes, containing gas atoms of the air. And these volumes change their positions by a very small amount, as soon as the pressure field is coming. We face the same situation of the electrons in the wire of the coil. We can imagine small finite volumes, containing electrons. And these volumes alter their positions by a small amount as soon as they are exposed to an electrical field.

The gas atoms of the air are moving very fast, same as the electrons in the wire. But the small finite elements of volume filled by electrons resp. gas molecules only move with a very moderate speed, which we know as the "drift-speed" in the case of the electrons and as the "acoustical velocity" in the case of the gas molecules. Nevertheless signals and waves are propagating with typical signals speed, which we know to be the "speed of sound" in acoustics and $v = 1/\sqrt{LC}$ in electrodynamics.

The signal speed defines the speed of propagation of the field and waves in the wire of the LC-oscillation-circuit. This means that we really control the speed of propagation of the interacting fields (of the electromagnetic interaction), when we adjust the inductivity "L" and the capacity "C" of the oscillation circuit.
This explanation demonstrates, how the DFEM-computation is traced back to the conversion of ZPE-energy, as explained in detail in [Tur 10a] and [Tur 10b]. It shall help the readers of this publication to understand, how the LC-oscillation-circuit is indeed used, to control the speed of propagation of the interacting fields, as being necessary for ZPE-conversion.

Now the principle our calculation is explained. The execution of the calculation can be seen in details in the algorithm in the source-code. And it is clear, that the calculation will give results. So, this is the time to begin to analyse the results. This is, to what we will turn our attention beginning with the next section.

3. Evaluation of the results of a converter example

A very first evaluation of the behaviour of the converter-system can be done with some first results, which the program displays on the screen, directly when the program is running. Table 1 is a list of the very first and important physical values for the evaluation of a ZPE-converter. The table is of course made for the example of our EMDR-converter, for which we later will have concrete hints how to build it up.

Physical entity	Explanation
$U_{cap,I,\max} = \frac{q_{I,\max}}{C_I}$	Maximum voltage the input-capacitor
$U_{cap,T,\max} = \frac{q_{T,\max}}{C_T}$	Maximum voltage at the turbo-capacitor
$\dot{q}_{I,\max}$	Maximum current in the input-coil
$\dot{q}_{T,\max}$	Maximum current in the turbo-coil
$L_I \cdot \ddot{q}_{I,\max}$	Maximum voltage of the input-coil
$L_T \cdot \ddot{q}_{T,\max}$	Maximum voltage of the turbo-coil
$\dot{\varphi}_{\max}$	Maximum angular velocity of the rotating magnet (rad/sec)
$\frac{\dot{\varphi}_{\max}}{2\pi}$	Maximum angular velocity of the rotating magnet (U/sec)
$E_{Anf} = E_{ges}(t=0)$	Initial start energy inside the system
$E_{End} = E_{ges}(t=Ende)$	Final energy inside the system at the end of the computation time
$E_{End} - E_{Anf}$	Increase of system-energy during the computation time
$\frac{E_{End} - E_{Anf}}{T_{ges}}$	Power due to the increase of system-energy during the computation time
$P_{ent} = \int_o^{T_{ges}} R_{Last} \cdot \dot{q}_T^2 \cdot dt$	Extracted energy at the load resistor
$\frac{P_{ent}}{T_{ges}}$	Average power being extracted at the load resistor
$E_{in} = \int_o^{T_{ges}} \dot{q}_I \cdot U_{in}\, dt$	Total energy being introduced by the input voltage
$\frac{E_{in}}{T_{ges}}$	Average power being introduced by the input power supply
$P_{mech} = M_{mech} \cdot \dot{\varphi}$	Mechanical power extraction by the torque $M_{mech} = c_r \cdot \dot{\varphi}$, with $c_r =$ coefficient of friction for power extraction proportional to the angular velocity.
T_{ges}	The relation of the analysis

Tab. 1: Overview of some results presented on the screen.

Several further physical values should be analysed due to their dynamic behaviour as a function of time. They are listed in table 2. The program exports these data into a file which can be read by Excel. There they are available for being displayed graphically. Thus table 2 also contains information about the column in which the data are to be found in Excel.

Excel-column	Physical size	How to calculate it
A	t	Time-scale
B,C,D	q_T , $\dot{q}_T$, $\ddot{q}_T$	Electrical charge and its derivations in the turbo oscillation circuit
E,F,G	q_I , $\dot{q}_I$, $\ddot{q}_I$	Electrical charge and its derivations in the input oscillation circuit
H,I,J	φ, $\dot{\varphi}$, $\ddot{\varphi}$	Angle of the rotating magnet and its time dependent derivations
K,L	ψ_I , ψ_T	Magnetical flux through the coils
M,N	$U_{ind,I}$, $U_{ind,T}$	Voltage induced into the coils
O,P	$E_{mag,I}$, $E_{mag,T}$	Energy within the coils: $E_{mag} = \frac{1}{2} \cdot L \cdot \dot{Q}^2$
Q,R	$E_{cap,I}$, $E_{cap,T}$	Energy within the capacitors: $E_{cap} = \frac{1}{2} \cdot C \cdot U^2 = \frac{1}{2} \cdot \frac{Q^2}{C}$
S	E_{rot}	Energy of the mechanical rotation: $E_{rot} = \frac{1}{2} \cdot J \cdot \omega^2 = \frac{1}{2} \cdot J \cdot \dot{\varphi}^2$
T	E_{ges}	Total energy in the system: $E_{ges} = E_{mag,I} + E_{mag,T} + E_{cap,I} + E_{cap,T} + E_{rot}$
U	E_{Last}	Power extracted by the load resistor: $E_{Last} = R_{Last} \cdot \dot{q}_T^2$
V	U7	Input voltage
W	P_{zuf}	Power being introduced by input power supply
X	c_r	Coefficient of friction, can be varied as a function of time
Y	P_{mech}	Mechanical power extracted by friction $P_{mech} = P_{reib} = M_{reib} \cdot \dot{\varphi} = c_r \cdot \dot{\varphi}^2$
Z	NULL	Auxiliary column

Tab. 2: Overview over the data being exported into Excel.

Now our DFEM-algorithm is developed so far, that we can perform realistic computations of arbitrary electric and/or magnetic ZPE-converters. An example therefore shall be calculated in the following sections - namely with the EMDR-converter suggested by the author of this publication.

4. Computation example for a concrete ZPE-motor

The namely reason for the development of the DFEM-algorithm presented here is the fact, that the author wants to show realistic computations on his EMDR-Converter („Electro-Mechanical Double-Resonance“ Converter), and to get reliable results with sensible precision on this machine. This has the purpose to prepare building up an experimental prototype. Even if the author of this article can not built up such a prototype by himself due to his very restricted possibilities, he hopes, that many colleagues will read this article and try to build up prototypes by themselves.

In order to make it most efficient for those colleagues, who already began to think about building up such prototypes, the author decided to develop the new calculations for a setup which is a rather similar to this one in his former calculation performed in [Tur 11]. The setup is, what we saw in figure 9.

The definition of the geometry in the computer program needs a set of the 32 input parameters. Additionally the program works with some constants of nature and some other parameters derived from the input parameters (additionally 18 such parameters), which are displayed on the screen during the runtime of the program. Sensible definition of the input parameters needs exact fine tuning of their values, and thus take several hours/days even when the author does it by himself.

Caution: The parameter-set displayed on the following pages corresponds directly to the geometry and setup of figure 9. Different design needs different parameters. Furthermore the program contains several subroutines, which are developed to perform automatic meshing for the finite element method. But this automatic meshing only works, if the setup is not altered by principle. If somebody wants to change the design of the converter, it will be necessary to change the subroutines as well.

(a.) Definition of the geometry of the setup

We start now with the explanation of the input parameters, which the realistic DFEM-algorithm requires. For didactical reasons we now do not want to take mechanical power extraction into consideration, because if we leave this aspect for later, it will be easier to understand the converter. Mechanical power extraction will be introduced later within this publication.

{Constants of nature, not Input-parameters:}
- epo:=8.854187817E-12{As/Vm}; {electric field constant}
- muo:=4*pi*1E-7{Vs/Am}; {magnetic field constant}
- LiGe:=Sqrt(1/muo/epo){m/s}; {speed of light}

{For the solution of the differential-equations and for the display of the results:}
- AnzP:=5000000; {number of time steps of the numerical iteration}
- dt:=1E-6; {Sec. } {Duration of each single time step}
- Abstd:=1; {only for preparation, do not alter the value}
- PlotAnfang:=0000; {For Data-export to Excel: First Plot-Punkt}
- PlotEnde:=5000000; {For Data-export to Excel: Last Plot-Punkt}
- PlotStep:=200; {For Data-export to Excel: step width of the data being exported}

{Remark: Excel is restricted to maximal 32.767 Data-groups. If the number of time steps is larger than this value, not all computed data can be plotted graphically by Excel. In our example, only every 200th point is being exported to Excel.}

{For the definition of the geometry of the coils (DFEM-meshing is done automatically):}
- Spsw:=0.01; {Meters of step width of the meshing}
- xo:=0; yo:=6; zo:=5; {Geometrical parameters according Fig.1, steps of Spsw}
- Ninput:=80; {number of windings of the Input-coil, left coil in figure .1}
- Nturbo:=12; {number of windings of the Turbo-coil, right coil in figure .1}
- nebeninput:=8; {windings side-by-side in Input-coil}
- ueberinput:=10; {windings on top of each other Input-coil}
- nebenturbo:=3; {windings side-by-side in der Turbo-coil}
- ueberturbo:=3; {windings on top of each other Turbo-coil}

{Remark: Here the parameters are used to define rectangular coils according to Fig. 1. The cross-section of the Input-coil consists of 8 windings side-by-side and 10 such layers on top of each other. The cross-section of the Turbo-coil consists of 3 windings side-by-side and 3 such layers on top of each other. “On top of each other”means, that the layers are built up radially.}

{For the emulation of the permanent magnet:}
- Bsw:=1E-2; {Meters} {The magnetic field shall be stored in steps of centimetres.}
- MEyo:=0.05; {Half length of the cylindrical bar magnet}
- MEro:=0.01; {Radius of the cylindrical bar magnet}

- MEI:=15899.87553475; {Amperes, current in the coil is to emulate the permanent magnet}
{Remark: We here use a cylindrical bar-magnet according to Fig. 8 and Fig.9. The shape can be altered if required, but the meshing-subroutines have to be altered also.}
{Remark regarding the data-storage of the magnetic field: The magnetic field is fixed rigidly to the magnet within a sufficiently extended volume of space. The values of the field strength are stored in a data-array at finite geometrical steps. When the magnet is moving, the field is moving together with the magnet. The step length for the data-storage of the field is „Bsw".}
{Remark: The weird value for the electrical current in the magnet-emulation-coils has its reason in the fact, that a given value of the magnetic field has to be emulated. The field strength of this field to be emulated is displayed on the screen during the runtime of the program. It can be read in order to adjust the current MEI in such way that the required field strength is achieved. In our example we have a magnet with 1 Tesla at its ends.}

{Further technical dimensions:}
- DD:=0.10; {Meter} {Thickness of the wire from which the coil is made}
- rho:=1.35E-8; {Ohm*m} {Specific electrical resistance of copper, [Koh 96]}
- rhoMag:=7.8E3; {kg/m^3} {Density of the magnet-material, Iron, [Koh 96]}
- CT:=36.61E-6; {Farad} {Capacity within the oscillation circuit of the Turbo-coil}
- CI:=100E-6; {Farad} {Capacity within the oscillation circuit of the Input-coil}
{Remark: In our example, the input-coil has been modelled in order to prepare it for everybody who wants to have an additional coil in the DFEM-algorithm. But the input-coil is not used for the computation of the converter, and the complete oscillation circuit containing the input-coil is left away for the computation of the self running EMDR-converter.}

- Rlast:=0.0111; {Ohm} {Ohm'ian load resistor in the Turbo-circuit for the extraction of energy}
- UmAn:=50000; {U/min} {Initial angular velocity: mechanical boundary condition - rotating magnet}
- Uc:=0;{Volt} Il:=0; {Ampere} {electrical boundary conditions – capacitor-voltage, current in the coil}
{Remark: Even a self running ZPE-motor needs an initial energy to be started. It will not start just by alone. The initial energy can be supplied mechanically (as it is done in the example here), but it can be supplied electrically as well, for instance by charging the capacitor the coil in order to initialize the operational the motor.}
- U7(t)=0;
{Remark: If the ZPE-converter is not a self running engine, but only an over-unity engine, it permanently needs some classical input-energy for operation. This can be supplied mechanically (at the axis of rotation) or electrically with some input-voltage. The last version is displayed in the source-code printed in the appendix, by the use of a subroutine with the name "U7". Nevertheless this input-voltage is not used for the solution of the differential-equations, because the EMDR-converter is a self running engine and does not need any classical input power.}

{Composed Parameters, for the purpose to control the input data. Do not use them for input.}
- DLI:=4*(yo+zo)*Spsw*Ninput; {Meter} {length of the wire of the Input-coil}
- DLT:=4*(yo+zo)*Spsw*Nturbo; {Meter} { length of the wire of the Turbo-coil }
- RI:=rho*(DLI)/(pi/4*DD*DD); {Ohm} {Ohm`ian resistance of the wire of the Input-coil}
- RT:=rho*(DLT)/(pi/4*DD*DD); {Ohm} { Ohm`ian resistance of the wire of the Turbo-coil}
- BreiteI:=nebeninput*DD; HoeheI:=ueberinput*DD; {Width and height of the Input-coil}
- BreiteT:=nebenturbo*DD; HoeheT:=ueberturbo*DD; {Width and height of the Turbo-coil}
- fkI:=Sqrt(HoeheI*HoeheI+4/pi*2*yo*2*zo)/HoeheI; {Induktivity-correction for short coil}
- fkT:=Sqrt(HoeheT*HoeheT+4/pi*2*yo*2*zo)/HoeheT; {Induktivity-correction for short coil}
- LI:=muo*(2*yo+BreiteI)*(2*zo+BreiteI)*Ninput*Ninput/(HoeheI*fkI);
{Geometrical average => Induktivity of the Input-coil}
- LT:=muo*(2*yo+BreiteT)*(2*zo+BreiteT)*Nturbo*Nturbo/(HoeheT*fkT);
{Geometrical average => Induktivity of the Turbo-coil}

- omT:=1/Sqrt(LT*CT); {Resonance-angular-frequency of Turbo-cicuit of LT & CT}
- TT:=2*pi/omT; {classical duration of oscillation of Turbo-circuit of LT & CT}
- Mmag:=rhoMag*(pi*MEro*MEro)*(2*MEyo); {Mass of the Magnet}
- J:=Mmag/4*(MEro*MEro+4*MEyo*MEyo/3); {moment of inertia of the magnet of rotation}
- omAn:=UmAn/60*2*pi; {Start angular velocity (rad/sec.) of the rotating magnet}
- UmSec:=UmAn/60; {Start angular velocity, rotating Magnet (rounds per second)}

{Remark: Some of these values are not only necessary to control the input data but they are also necessary for the further computation in the DFEM-Algorithm.}

With these parameters, the ZPE-motor is modelled as being displayed in figure 11. As explained above, there is only the Turbo-coil and no Input-coil.

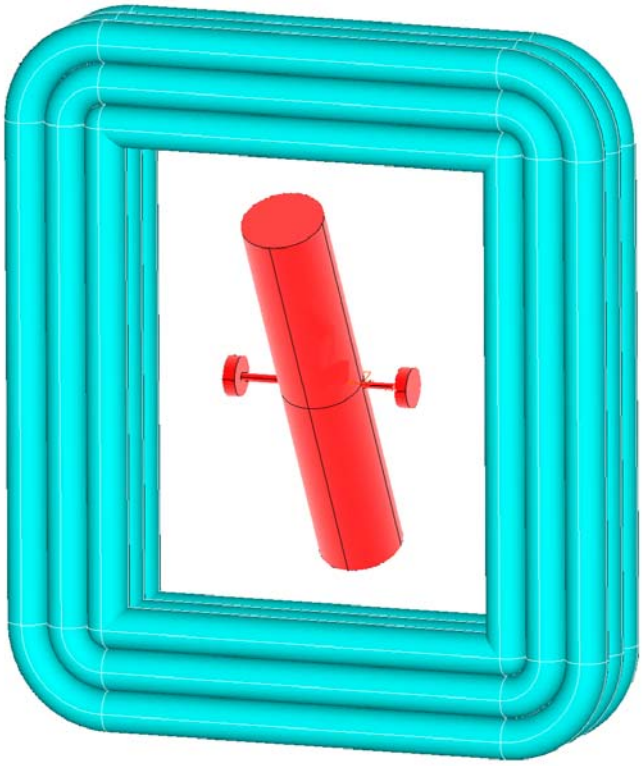


Fig.11:
(Drawn by the use of ANSYS [Ans 08].)

EMDR-converter with a rotating magnet (red colour: length 10 cm, thickness 2 cm).

Also in red colour we see a thin axis of rotation, around which the magnet can rotate. At each end of the axis, there is a bearing (also red) to keep the axis of the rotating magnet at its position.

The windings of the Turbo-coil are drawn in light blue colour. As we see, the field flux lines of the rotating magnet cut the windings of the coil exactly perpendicular.

Because of practical aspects (as can be seen later), the windings of the coil have to be made from rather thick material, and the coil should be made of not very many windings. In our example, the cross-section of the coil has an inside area of 5 × 6 cm^2, and the thickness of the wire 10 mm.

The time steps for the numerical iterative the solution of the differential-equations have to be choosen fine enough, that every period of motion consists of sufficiently many steps of calculation. Thefore the parameters as displayed above have time steps of $dt = 1\mu\sec.$ at an angular velocity of 50,000 rpm. This is for sure not sufficient. The time steps must be much smaller.

But during the practical application of the DFEM-algorithm, we start with rather rough time steps in order to get a first rough feeling for the behaviour of the machine. Rough time steps save computer (CPU)- time. We can use such rough times steps, to see what is happening, during the phase of definition of the geometry of the machine. We can also use such rough time steps for the maximisation of the converted power, when we want to alter the system-parameters by hand. As soon as this task is done, we have to reduce the time steps remarkably in order to check the convergence of the algorithm. Good convergence is achieved, when we see that any further reduction of the length of the time steps does not have a remarkable influence on the results. As soon as we reach the condition of good convergence, we have to begin the fine tuning of the system parameters, together with a renewal of the maximisation of the converted power.

In order to make the optimisation of the system parameters as efficient as possible, there is a data-output routine included into the program. This allows a fast evaluation of the behaviour of the system, when we alter the system parameters by hand (by trial and error). During this phase, we see the following data on the screen:

- The start energy within the system, which is brought into the differential-equations by initial conditions. This energy is computed among others from the parameters UmAn, Uc, Il.
- The energy within the system at the end of the time of analysis. Therefore all amount of energy within the system summed up, this is the energy of motion of the magnets, the energy within the coils and the energy within the capacitors.
- The amount of energy being gained within the system during the analyzed time of operation. This is the difference between the start energy and the energy at the end of the analysis-time. If the energy-gain plus the energy being extracted from the system is positive, the system converts ZPE-energy into classical energy. If this energy-gain plus the energy being extracted is negative, the system converts in the opposite manner classical energy into ZPE-energy. An ideal classical electromotor will contain the same start energy plus input energy, as it contains energy at the end of operation plus energy being extracted during operation. (Friction to be taken into account as energy-extraction.)
- The power corresponding to the energy gain, which is calculated by dividing the energy gain trough the time of observation.
- The energy being extracted from the Ohm'ian resistor during the time of operation, and the power corresponding with this energy gain. Not very interesting is the knowledge about the Ohm'ian losses in the wires of the coil. These losses are included into the calculation, but they are not printed on the computer screen, because we do not have any influence on them.
- The average of the mechanically extracted power and the sum of the mechanically extracted energy during the time of observation. Mechanical energy and power can be extracted from the rotating shaft (which is located in the middle of the permanent magnet). Mechanical energy is not yet regarded now in the sections 4 and 5, but soon in section 6 it will be regarded detailed.
- Additional to the mechanically extracted power and energy, there is some mechanical gain of energy, which remains inside the system.
- The input-voltage and the input-power in connection with the input-supply is printed on the computer screen, but it is ZERO, because there is no classical energy brought into the system during operation. The EMDR-converter is a self running system. The values are only displayed for those, who want to modify the DFEM-algorithm to simulate an over-unity machine.
- The total duration of the observation, which is the sum of all time-steps "dt". We do not speak about the elapsed CPU-time, but we speak about the simulated time of operation during which the converter is running.

The simple online data-evaluation as described above helps the user to get a quick impression about the mode of operation of the converter, which allows to perform a variation and optimization of the system-parameters by hand. This means, that we can alter the values of the system parameters, check the behaviour of the system, alter the values again, check again, and so on… By this means we should be capable to develop a design which should work properly (at least theoretically).

As soon as this design is found, it is recommended to evaluate the system more precise. This can be done by observing those variables as a function of time, which follow the dynamics of the motion. Therefore the time dependent variables of the system are exported into a data-file for Excel, where

they can be displayed graphically. The listing of the contents of the different Excel-columns have been printed above in Table 2. From the evaluation of these data, we have to answer questions such as the following:

- Does the system run into a stable mode of operation ?

  This can be seen for instance, when we regard the angular velocity $\dot{\phi}$ as a function of time, because a stable mode of operation can only be achieved if the angular velocity is constant, or at least if it oscillates around a constant value (within a well controlled hysteresis). If this is not the case, the converter is still in the phase of initialisation, or it does not run stable at all. In order to decide between these both possibilities, the length of the time steps “dt” should be reduced, and the total oberservation time has to be enhanced. Then the calculation shall be repeated with longer observation time.

  The same observation can be done with regard to the electrical current in the coil or with the voltage in the capacitor.

- Does the machine convert enough ZPE-energy, so that it will not be brought to standstill by friction ?

  Therefore we check the energy of the mechanical rotation remaining in the rotation of the permanent magnet, as a function of time. The gain should be large enough, that we can expect, that it is sufficient to surmount the energy loss due to friction. If the calculation is done taking mechanical power extraction into account (see section 6), we can define the coefficient of friction and check the extracted power (as a function of time).

- Also the time dependent behaviour of the electrical currents and the voltages in the LC-oscillation circuit should be observed graphic only, because this contains more reliable and detailed information than the simple estimation of the maximum, which is printed during the runtime on the screen. If the machine is running properly as a ZPE-converter, the values of the voltage and the current are nomaly rather large, so that we have to be careful not to overload the wire of the coil or the capacitor.

By the way it should be mentioned, that the elapsed CPU-time can take several minutes or several hours, especially when the time steps are very short (for instance in the range of few nanoseconds).

5. A concret EMDR vacuum energy converter

The DFEM-program in the appendix can only run, if there is a data-file in the same directory with the name *'schonda'*, which can be downloaded together with source-code of the DFEM-program for free from the Internet-page of the author of the publication presented here. The data-file *'schonda'* only has the purpose to save CPU-time, as following: During the phase of initialisation of the main program, all preliminary work takes some CPU-time, which is only necessary, if the parameters describing the geometry of the setup have been altered since the last run of the program. (In this case, the automatic meshing has to be renewed.) If the geometry-parameters remain unchanged since the last run of the program, the results of the initialisation can be taken directly from the last run of the program (together with the existing mesh, stored in ‘*schonda*’). This is exactly what the program does, when it reads the data-file *'schonda'*, in order to save the time for the same initialisation which already has been calculated before. This makes it more efficient to repeat the program several times during the phase of the optimisation of the system parameters.

We now (in section 5) want to discuss the system parameters, with the values as printed in the source-code in the appendix.

The typical construction of a EMDR-converter begins with a search for an appropriate permanent magnet. As soon as the magnet is found and its field strength is measured (for instance with a Hall-probe), its dimensions are put into the input-pata lines of the source-code. Then we have to construct the conductor loops for the emulation of the permanent magnet and to apply an appropriate electrical current within the conductor loops, in order to reproduce the measured values of the magnetic field.

Then we have to modulate the Turbo-coil and to put all the other requested data into the DFEM-algorithm. Finally the initial angular velocity of the rotating magnet has to be adopted, and the very last step is the adjustment of the capacitor in the Turbo-oscillation-circuit. The criterion of the initial angular velocity of the rotating magnet shall be decided from the stability of the bearing keeping the rotating axis. The bearing must withstand the angular velocity of the stable mode of operation, which can be remarkable larger than the initial angular velocity. Thus the value of the maximum possible angular velocity of the magnet plays an important role for the adjustment of the capacitor. High-speed rotation has the consequence to enhance the amount of power being converted from the ZPE of the quantum vacuum. If the system operators as a ZPE-converter, the angular velocity is increasing during the initial phase of the operation. And we can use the amount of the increase of the angular velocity as an indicator for the quality of adjustment of the system parameters. The more ZPE-power we convert, the more increase of the angular velocity we observe. At the very beginning of the adjustment procedure, we apply a very small load resistor and adjust the capacity to a maximum of increase of the angular velocity of the magnet.

Rather often (depending on the geometry of the system) we observe, that the mechanical power-gain is much larger than the electrical power-gain. This is also the reason, that we will soon (in section 6) have to take the real benefit of the converter mechanically from the rotating shaft.

The next step of the optimisation now consists in enhancing the load resistor in many small steps, and always readjusting the capacitor, while checking the mechanical and electrical power gain. Depending on the configuration of the system parameters, the mechanical and the electrical power can increase both or decrease both at the same time, but it is also possible that one is increasing and the other one is decreasing. The load resistor should not be enhanced to much, otherwise it will attenuate the LC-circuit rather strong, and thus prevent the engine to start properly.

If the mechanical power gain of the machine is rather large, this is absolutely no problem, because it simply indicates that we have enough mechanical power, to overcome friction without any problems. The way how to extract mechanical power will be the topic of section 6.

Even the theoretical adjustment procedure makes clear, that the load resistor as well as the capacitor have to be adjusted with very high precision. The consequence is, that we need a capacitor and a load resistor for the practical setup, which can be adjusted very precisely. The precision of the adjustment-quality should be at least somewhere between 1% and 0.1%. When we will soon see, how large the voltage and the electrical current in the LC-oscillation-circuit really are, we will understand that this defines a really serious requirement to the *capacitor* and to the *resistor*.

The example under discussion here is not optimized with regard to the converted power, because of this optimisation will be much better soon in section 6. Nevertheless it should be mentioned, that the converted power can be enhanced remarkably, when we enhance the number of windings in the Turbo-coil a little bit (accepting the disadvantage, that the voltage in the capacitor will be enhanced strongly if we do so).

The most effective way to enhance the converted power is the angular velocity in the stable operation. Enhancing the angular velocity has the consequence to enhance the converted power tremendously. 30,000 rounds per minute as used in the example here is not a really high angular velocity. Such a rather moderate value has the purpose to make it easier to find an adequate bearing for the rotating permanent magnet. But if we have the typically angular velocity of Turbo-rotor in the

automotive industry in mind (which spins with about 100,000 rounds per minute or even more), we see that an enhancement of the angular velocity should not be very difficult - and with it a remarkable enhancement of the converted power. Even much higher angular velocity is known from dentistry and from turbomolecular vacuumpumps in ultra high vacuum technology. If we could use such speedy rotation, the restriction to the power density of the ZPE-converter should probably be the electrical current in the copper wire, which should not be too strong, so that the copper wire will not get too hot.

If the results of the algorithm shall be documented, there is a simple option to press the "D"-button before you leave the program with the last <wait>, and the program will write the input-data as well as the most important results into a file named „Auswertung". This file can be read with a text program. An example for such a file is printed here.

DFEM-Simulation of an EMDR-Motor (here still without mechanical power extraction)

Parameters for the solution of the differential equation and the output of the results:
AnzP = 10000000 {number of time steps for the observed operation}
dt = 2.000E-0007 {seconds, duration of each time steps for the solution of the differential equations}
Abstd= 1 {only for preparation, do not alter the value}
PlotAnfang = 0 {first point for the Data-export to Excel }
PlotEnde = 10000000 {first point for the Data-export to Excel }
PlotStep = 400 {step width for the Data-export to Excel }

{Definition of the both coils (turbo and input):}
Spsw = 0.010000 {Meters: step width for the automatic mesh-generation of the coils}
xo = 0, {number of steps of Spsw}
yo = 6, {number of steps of Spsw}
zo = 5, {number of steps of Spsw}
Ninput = 100 {number of windings of the input-coil}
Nturbo = 9 {number of windings of the turbo-coil}
nebeninput = 10 {windings side-by-side of the input-coil}
ueberinput = 10 {layers of windings of input-coil}
nebenturbo = 3 {windings side-by-side of the turbo-coil}
ueberturbo = 3 {layers of windings of the turbo-coil}

Bsw = 1.0E-0002 {spatial resolution of the computation of the magnetic field}
MEyo = 5.00000E-0002 {y-coordinates of the conductor-loops for the emulation of the permanent magnet}
MEro = 1.00000E-0002 {Radius of the conductor-loops for the emulation of the permanent magnet}
MEI = 1.58998E+0004 {Electrical current in the conductor-loops for the emulation of the permanent magnet}

further technical dimensions:
DD = 0.0100000 {Meters} {diameter of the wire for the coils}
rho = 1.35000000000000E-0008 {Ohm*m} {Specific electr. resistance of copper, depending on temperature}
rhoMag = 7.80000000000000E+0003 {kg/m^3} {density of the permanent magnet, Iron}
CT = 9.83000E-0005 {Farad} {capacitor in the turbo-circuit}
CI = 1.00000E-0004 {Farad} {capacitor in the input-circuit}

additional necessary values:
Rlast = 6.400000E-0002 {Ohm} {Ohm´s load resistor in the LC-Turbo-circuit
UmAn = 30000.00 {U/min} {mechanical initial conditions - rpm of the rotating magnet}
Uc = 0.00 {Volt} {electrical initial conditions - voltage at the turbo-capacitor}
Il = 0.00 {Ampere} {electrical initial conditions - electrical current in the turbo coil}

composed parameters. These values are calculated, no input possible:
DLI:=4*(yo+zo)*Spsw*Ninput = 44.00000 {Meter, length of the wire oft he Input-coil}
DLT:=4*(yo+zo)*Spsw*Nturbo = 3.96000 {Meter, length of the wire oft he turbo-coil}
RI:=rho*(DLI)/(pi/4*DD*DD) = 0.00756 {Ohm} {Ohm`resistance of the wire of the Input-coil}
RT:=rho*(DLT)/(pi/4*DD*DD) = 0.00068 {Ohm} {Ohm`resistance of the wire of the turbo-coil}
BreiteI:=nebeninput*DD = 0.10000 {Width of the input-coil}

HoeheI:=ueberinput*DD = 0.10000 {height of the input-coil}
BreiteT:=nebenturbo*DD = 0.03000 {Width of the Turbo-coil}
HoeheT:=ueberturbo*DD = 0.03000 {height of the Turbo-coil}
fkI:=Sqrt(HoeheI*HoeheI+4/pi*2*yo*2*zo)/HoeheI = 123.61179 {factor of correction for inductivity}
fkT:=Sqrt(HoeheT*HoeheT+4/pi*2*yo*2*zo)/HoeheT = 412.02703 {factor of correction for inductivity}
LI:=muo*(2*yo+BreiteI)*(2*zo+BreiteI)*Ninput*Ninput/(HoeheI*fkI) = 1.24238647244960E-0001 {Induktivity, Input-coil}
LT:=muo*(2*yo+BreiteT)*(2*zo+Breitet)*Nturbo*Nturbo/(HoeheT*fkT) = 9.93606632469255E-0004 {Induktivity, Turbo-coil}
omT:=1/Sqrt(LT*CT) = 3.19974964955735E+0003 {classical angular frequency of the Turbo-circuit}
TT:=2*pi/omT = 1.96364903362012E-0003 {classical period of the Turbo-circuit }
Mmag:=rhoMag*(pi*MEro*MEro)*(2*MEyo) = 0.245 kg {Mass of the Magnet}
J:=Mmag/4*(MEro*MEro+4*MEyo*MEyo/3) = 2.10329628157837E-0004 {moment of inertia of the rotating magnet}

Several parameters, to be calculated from the above values:
Magnet: start angular velocity.: omAn = 3141.592654 rad/sec
Magnet: start angular velocity, Umdr./sec.: UmSec = 500.0000000000 Hz
Mass of the Magnet = 0.245044 kg
moment of inertia of the rotating magnet: 2.10329628157837E-0004 kg*m^2
total duration of observation: 2.00000000000000E+0000 sec.
Excel-Export: 0.00000E+0000... 2.00000E+0000 sec., Step 8.00000E-0005 sec.
These are 25000 data-point tob e exported to Excel.

****

Some of the results of the computation:
initial energy within the system: 1037.93511187 Joule
final energy within the system: 1171.18167853 Joule
power gain in the system: 66.62328333 Watt
energy extracted at the load resistor = 13.29139772 Joule
power extracted at the load resistor= 6.64569885828765E+0000 Watt
inserted energy by voltage supply: 0.00000000000000E+0000 Joule
inserted power by voltage supply: 0.00000000000000E+0000 Watt
total duration of the observation 2.00000000000000E+0000 sec.

This is the documentation of a set of important data written automatically by the program. We now want to discuss these data and further more other data:

The duration of observation is 2.0 seconds (the elapsed CPU-time is much larger on a normal computer). During this time-interval, the system gains a mechanical power of 66.62 Watts, and additionally an electrical power of 6.645 Watts. For both of them, there is no classical source of energy, so the energy can only come from some non-classical source which we regard as invisible for classical eyes. According to former explanations, it should be the energy of the electromagnetic zero-point waves of the quantum vacuum.

If we enhance the duration of the observation, we see that the converted power is reduced. The reason is not an incomplete convergence of the algorithm (for instance due to the duration of the finite time steps “dt”), but the reason is the fact, that the engine has totally different behaviour within the initial phase then after reaching a stable equilibrium mode of operation. We see this when we regard figure 12, which displays the angular velocity of the rotating magnet as a function of time (this is column ”I” in the Excel-data, of the automatically written file with the name „test“.) Further explanation will follow later.

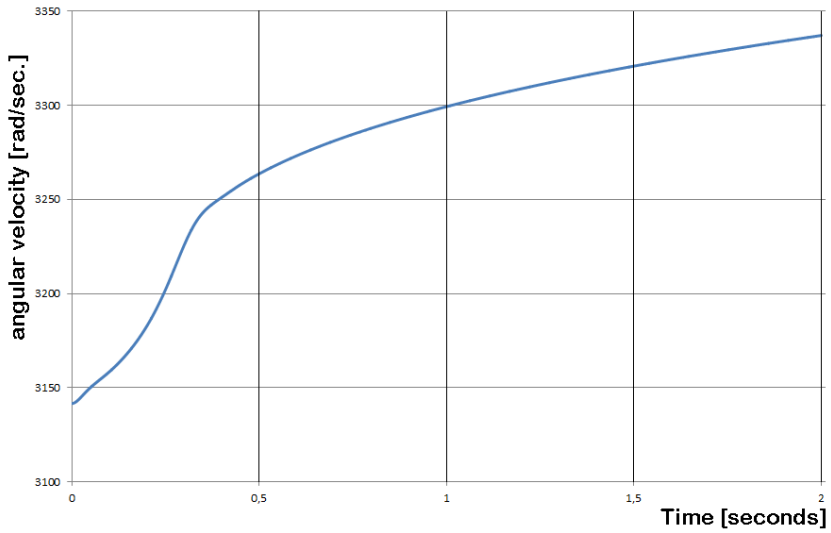


Fig.12:

Angular velocity of the rotating magnet, which increases as function of time.

At the begin of the initial phase, the energy gain is rather large. After some time the operation converges to saturation, but within the two seconds under analysis, it does not yet reach the saturated stable condition.

The voltage of the capacitor can be calculated from the electrical charge inside the capacitor as being plotted in figure 13. Please notice, that the calculation was done with 10.000.000 time steps, which is far too much to be resolved by any computer graphics. Consequently we only can see the envelope of the oscillation.

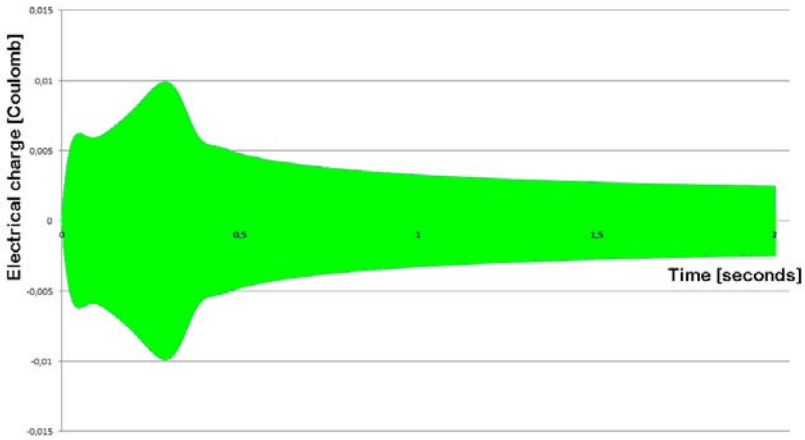


Fig.13:

In green colour we see the envelope of the oscillation of the electrical charge, which is flowing into and out of the capacitor.

The maximum of the voltage-amplitude is to be found at $U_{C,\max} = \frac{Q_{\max}}{C_{Turbo}} = \frac{0.01C}{98.3\mu F} \approx 102 Volt$.

It should not be a problem to get such capacitors. If the number of windings is enhanced, it is possible to work with higher voltage, which allows to enhance the electrical power in the circuit as well as the power ready for extraction (at load-resistor as well as mechanically).

The current in the coil, shown in figure 14, has a maximum of a bit more than 30 Amperes. If the number of windings (of the Turbo-coil) is enhanced and the load resistor is adjusted together with other systems parameters (such as the capacitor) at the same time, a strong enhancement of the current is possible. With our wire according to figure 11, with a diameter of 10 mm (cross-section of 78.5 mm²), a current of 30 Amperes should not be a problem at all.

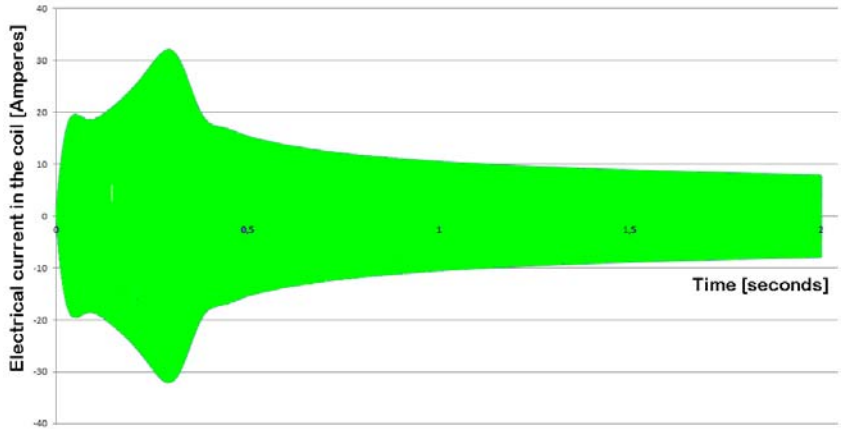


Fig.14:

Envelope of the oscillating current in the coil as a function of time.

The voltage over the coil, which graphically has a shape very similar to figure 13 and 14, should have an amplitude of the same order of magnitude as the voltage of the capacitor, as soon as the system is adjusted sensible. In our example we find a value of 99 Volts, which fulfils this criterion surprisingly well.

The angular acceleration of the rotating magnet, of which we see the envelope in figure of 15, displays a remarkable oscillation within each period of rotation. This makes it clear, that there are remarkable Lorentz-forces between the magnet and the coil of the LC-circuit. This indicates, that those colleagues who want to build up the design in a practical experiment, should take care of mechanical forces, which should be done by a stable fixation of the coil and the axis of the magnet.

A stable mode of operation is achieved, as soon as the oscillation of the torque (and the angular acceleration) is symmetrically around the abscissa - which is not the case here in our example, as can be expected from the discussion of figure 12. As we see, even at the end of the observation-time, the average of the angular acceleration is clearly positive, so that the rotor did not yet come to its maximum (final) angular velocity.

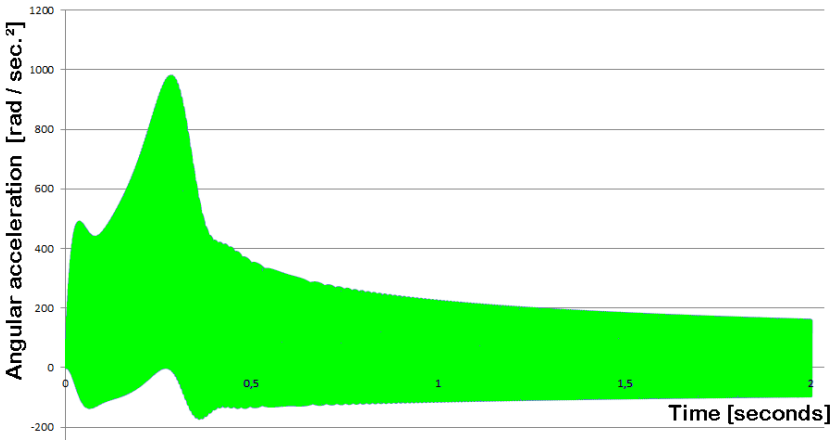


Fig.15:

Envelope of the angular velocity of the rotating magnet as a function of time.

The question about the quality of the numerical iteration can be answered, when we focus the plot to a high-resolution interval of time, so that we do not only see the envelopes of the curves, but the curves themselves. This is shown in figure 16, showing that time-zoom of the total duration of about 5 milliseconds, within which we can resolve about five periods of oscillation. The plot displays the angular acceleration. If the curves are irregular or not smooth, we have the typical case of too low scanning frequency (too long time steps “dt”), so that we face the necessity to reduce the length of time steps. Of course such an enhancement of the precision of the calculation does enhance the elapsed computer-time for the computation. In our example of figure 16, we see satisfactory numerical scanning, because the curve is a regular and smooth.

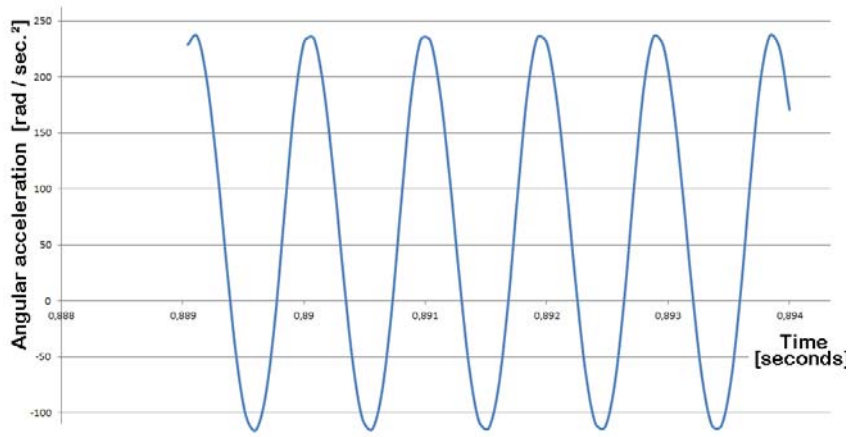


Fig.16:

The observation with high time-resolution allows us to evaluate the quality of the numerical iteration.

Although we permanently extract power from the system (see figure 17), the total energy within the system is increasing strictly monotonously during the whole time of our observation (see figure 18). This will be the case as long as the rotor does not achieve the stable equilibrium-condition of constant angular-velocity. Remark: The situation is totally different, as soon as we extract mechanical power from the system, but we will see this later in section 6.

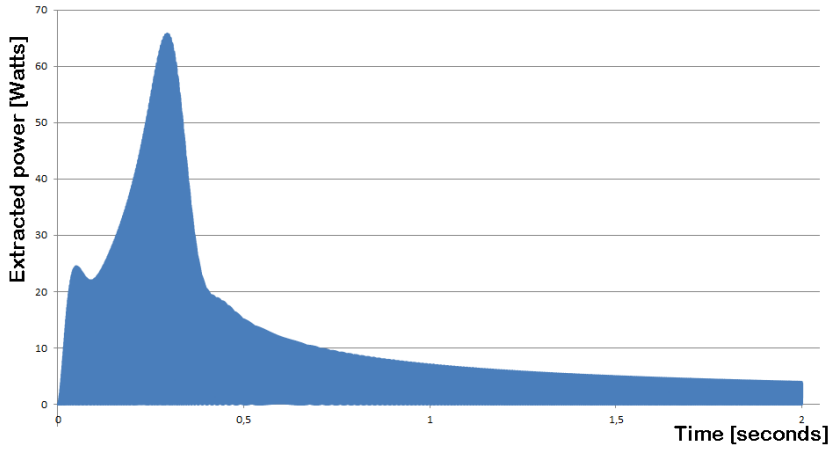


Fig.17:

Electrical power extraction at the load resistor.

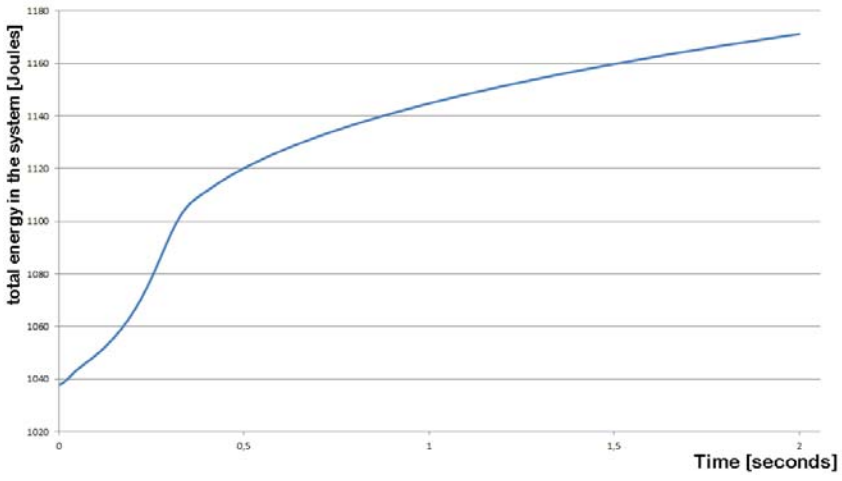


Fig.18:

Total energy within the system.

This is the energy of the coils, the capacitors and the rotation of the magnets.

In our example we do not have to analyse any power being supplied to the system, because the input-circuit is not existing at all, for there is no input of energy. The machine is a self-running engine (not an over-unity system).

A long-term analysis of the electrical power being extracted from the load resistor is shown in figure 19. The only aspect which was changed with regard to the short-term analysis reported before, is the

total duration of the observation, which is now 40 seconds for the computation to figure 19. Obviously the extractable power at the load resistor converges to ZERO. This observation is confirmed very clearly, if we perform a further enhancement of the duration time of the observation. This arises of the question, whether the EMDR-system is capable as a ZPE-motor at all !

The answer will be **YES** (!), as we will soon see in section 6. The EMDR-motor needs some mechanical resistance (such as for instance friction or power extraction) that it can work as a ZPE-motor.

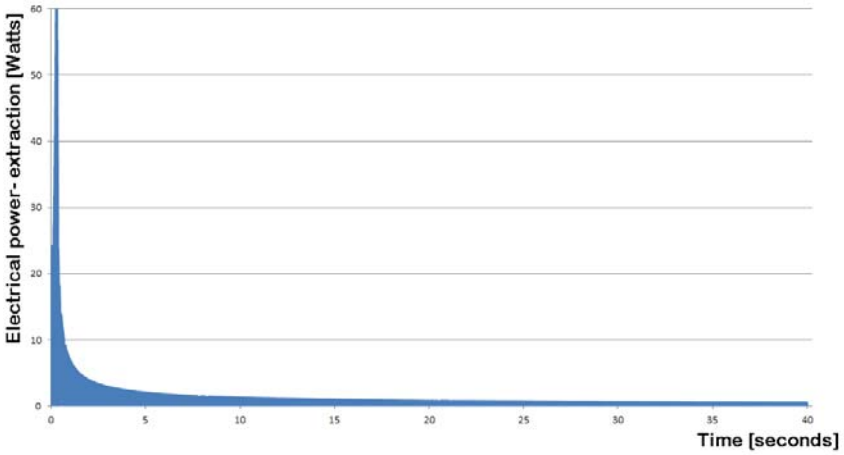


Fig.19:

Electrical power being extracted from the load-resistor in the Turbo circuit in a long-time analysis.

Let us discuss this now:

If we want to have the EMDR-converter in powerful operation, the machine needs a constant phase-difference between the electric oscillation in the LC-circuit and the mechanical rotation of the permanent magnet. There is an optimum value for this phaseshift, which has to be maintained during long-term operation, in order to maintain the conversion of ZPE-energy constantly. This phase shift is absolutely necessary, that the oscillating magnetic field made by the turbo-coil can accelerate the rotating magnet exactly in this moment, in which the turbo-coil-field is the most strong. But on the other hand the turbo-coil-field should decelerate the magnet at this moment at which it is the most weak.

Let us remind, that a constant field would accelerate and decelerate the magnet to the same amount, so that the angular velocity will alter during each period of rotation, but it will become the same after a full period of rotation. Differently from this, an oscillating field has the possibility to make the acceleration different from deceleration.

We can illustrate this explanation by a rather simple model, as following: The oscillation within the LC-circuit causes the energy within this circuit move periodically back and forth between the (energy of the) electrostatic field in the capacitor and (energy of the) the magnetic field in the coil. This is exactly happening twice per period, because the capacitor plates will be charged alternating “positive” and “negative”, as well as the coil is alternating between “north” and “south” (at each side). At these moments, in which the field energy is inside the coil (i.e. the field energy is the energy of a magnetic field), the magnet has to be accelerated remarkably. But in the moment when the field is inside the capacitor (i.e. the field energy is the energy of an electrostatic field), the magnet is decelerated by the coil only very weak, because the coil has almost no magnetic field.

A graphical illustration can be seen in figure 20 and 21, which is an animation of eight pictures cyclically following one after each other as a function of time. Let us start our considerations with figure 20, where we see the case of a “beneficial phase-shift”, driving a strong rotation. The capacitor is drawn with purple colour, and the electrical charge within the capacitor is illustrated by “positive” and “negative” algebraic signs noted at the capacitor-plates. The more charge we find inside the capacitor plates, the more algebraic signs are noted. When the field (and some electrical charge) moves into the oscillation-circuit, an arrow in black colour symbolises the direction of propagation of the field. Obviously the field has to pass the long wire, which is formed as a coil drawn in blue-colour.

This has the consequence, that there is an electric current within the coil, producing a magnetic field. This magnetic field is noted next to the wire of the coil by the use of the symbols („N“ and „S“). Again the number of symbols represents the strength of the field.

Let us now begin our considerations with part “1” of figure 20. This is the moment t_1, where the capacitor still contains some electrical charge, but it is far away from being charged to its maximum. This means that some of the field’s energy is in the coil, causing some magnetic fields (also away from its maximum). But we see the polarity of the magnetic field and the polarity of the permanent magnet, which are orientated relatively to each other in such a way, that the magnet will begin to rotate clockwise. In part “2”, the capacitor is discharged completely, so that the electrical current within the coil reaches its maximum. Now the total field’s energy is the energy of a magnetic field, so that the attractive force accelerating the permanent magnet is rather strong. This means that the angular velocity of the rotation is enhanced remarkably (because the phase-difference between the fields in the LC-circuit and the position of the rotating magnet is “beneficial”). In part “3”, the magnetic fields of the coil is reduced partially, but due to the orientation of the permanent magnet, there is no more torque and thus no angular acceleration on the magnet. In this situation, the magnet is simply continuing his rotation constantly. In part “4”, there is no more magnetic field in the coil, because there is no electrical current and the coil. This means that there is no deceleration which might like to move the magnet back with its “south-pole” to the position where we formally had the “north-pole” of the coil. In this situation, the magnet is also continuing his rotation constantly. But where is the energy of the oscillating field in the LC-circuit ? It is inside the capacitor, so that it does not have the chance to have any influence on the motion of the magnet. This means, up to now we can say, that the magnet was accelerated but not decelerated. In part “4”, the capacitor is charged up to its maximum. From now on the capacitor begins to discharge, which can be ovserved clearly in part “5”. Of course the discharge-current has the opposite direction as in part “1”, so that the magnetic field of the coil in part “5” has the opposite polarity as in part “1”. And this is fine, because the magnet also has the opposite direction as in part “1”. This means that the field of the coil begins to accelerate the rotation of the magnet also clockwise in part “5”. In part “5” we have a magnetic field growing slowly, but in part “6”, this magnetic field already reaches its maximum value, so that we have a good angular acceleration of the magnet which is again orientated clockwise. (The situation of “6” corresponds to the situation of “2”.) Part “7” now corresponds to the situation of “3”, and after the above explanation it is clear, that neither in “7” nor in “8” there is any deceleration disturbing the rotation of the magnet. The rotation is going clockwise, so that part “8” is followed by part “1”, and so on…

The acceleration of the magnet is active as long as the phase shift between the orientation of the magnet and the magnetic field in the LC-circuit is present.

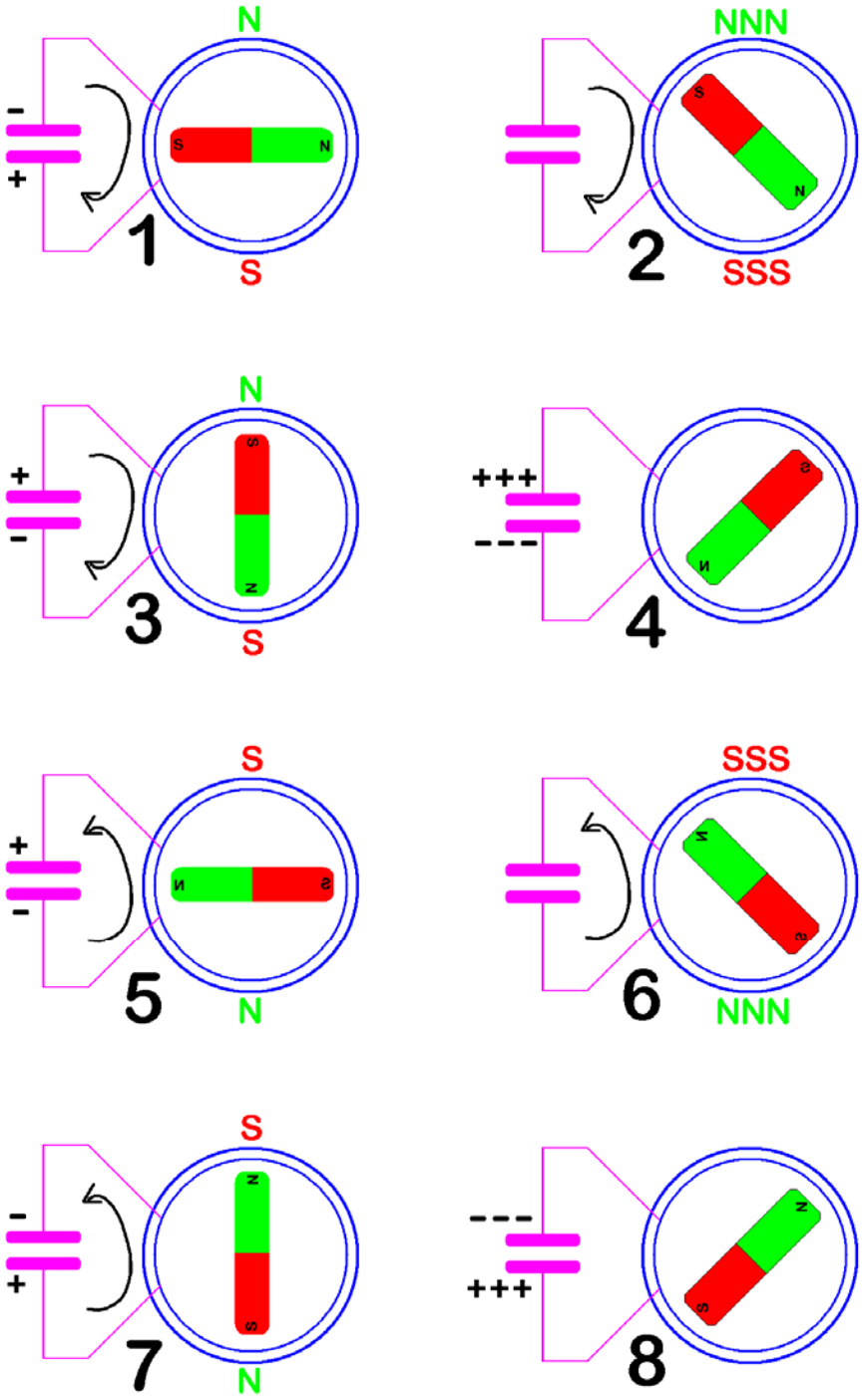


Fig.20:

Rotation of a magnet in the coil of a LC-circuit with constant "beneficial phase-shift" between the orientation of the magnet and the orientation of the oscillating magnetic field in the circuit.

If the magnet would be accelerated permanently, due to the beneficial phase shift, the angular velocity of the rotation would increase until the rotation of the magnet catches the rotation of the LC-circuit. But this would have the consequence, that the phase-shift between the electric LC-oscillation and rotation would disappear. And then the phase-shift would no longer be "beneficial", but it would come into a condition which we can see in figure 21. Under this condition there is no more acceleration of the magnet (i.e. no further conversion of vacuum-energy).

Let us begin our explanations with part "1" of figure 21. Here we still see a slight clockwise acceleration acting on the permanent magnet. The magnetic field of the coil is not extremely strong, but it is existing. Part "2" does not need very much explanation, because the orientation of the permanent magnet does not allow any acceleration at all. But interesting is now part "3" with counterclockwise acceleration of the permanent magnet. And as we see, the absolute value of the deceleration in "3" is of the same size as the absolute value of the acceleration in "1". This means

that the acceleration in “1” is completely compensated by the deceleration in “3”, indicating that the time from “1…3” does not cause any acceleration or deceleration in sum at all. Part “4” does not change our general situation, because here we again have no acceleration or deceleration. There is simply no magnetic field of the coil in part “4”. The train of thoughts, which we applied from “1…4” can be repeated for “5…8” analogously, so that we come to the conclusion, that during one turn, the angular velocity of the rotating magnetic is increasing and decreasing periodically, but there is no sum acceleration or deceleration at all. There is no “beneficial” phase-shift between the rotation of the magnet and the oscillation of the LC-circuit.

From here we understand, that the EMDR-converter permanently needs a “beneficial phase-shift” between the electrical and magnetic motion. If this phase shift decreases to “zero”, it is not further possible to convert any ZPE-energy. This means that we need the “beneficial phase-shift” for proper operation of the ZPE-motor. But the phase shift cannot be generated electrically (as for instance by a load-resistor). Thus we have to generate the “beneficial phase-shift” mechanically by applying some mechanical load to the rotating shaft at the axis of the magnet.

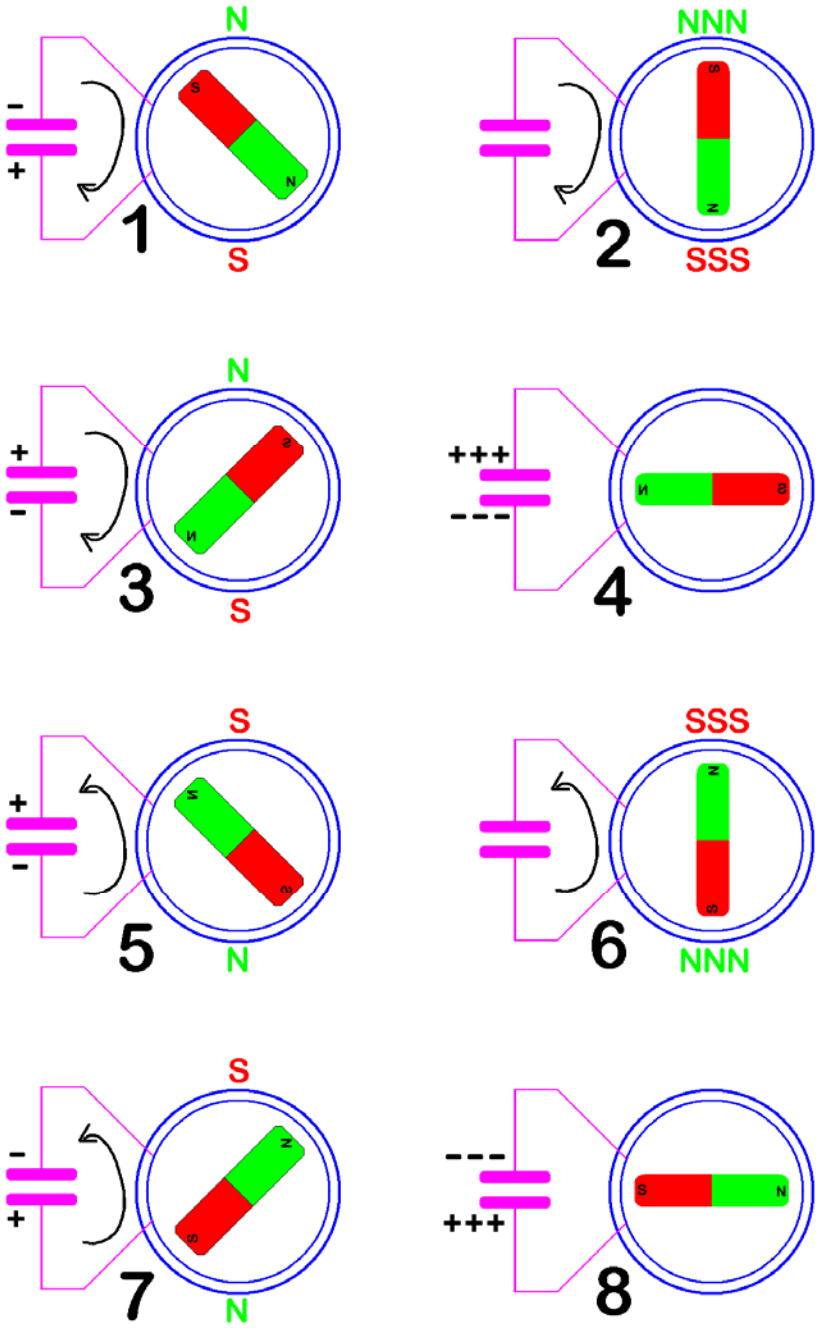


Fig.21:

Rotation of a magnet in the coil of a LC-circuit without “beneficial phase-shift” between the orientation of the magnet and the orientation of the oscillating field in the circuit.

Due to the absence of mechanical load, the rotating magnet can catch the position of the oscillating field, so that no further ZPE- energy will be converted.

This way of operation brings us to the knowledge, that the EMDR-converter can be operated as a ZPE-converter only, if permanently some mechanical power is extracted.

The fact that the rotating magnet can never overtake the oscillation of the LC-circuit (even not if it is free from any load), assures that there is an upper limit to the angular velocity of the mechanical rotation. This prevents us from the danger of too speedy rotation, which might cause a damage of the machine or an accident. Of course we have to make sure, that the rotating magnet can withstand the frequency of the LC-circuit.

In order to convert ZPE-energy, the EMDR-motor must have a positive "beneficial phase-shift", of which the ideal case is shown in figure 20, representing a phase-shift of 45°. For it is not possible to maintain this phase-shift only by electrical means, mechanical power extraction from the shaft is absolutely necessary. This means that the rotating axis of the permanent magnet has to be loaded permanently with some mechanical torque, in order to maintain the requested "beneficial phase-shift".

With regard to this aspect, our ZPE-motor shows completely different behaviour than a classical electrical motor. If we for instance decide to use good bearings in order to minimise friction and to avoid mechanical load to the shaft, we will even not be able to get any electrical power of the engine. But already if we begin to enhance mechanical friction in the bearings, this would help to enhance the amount of electrical power which can be extracted.

Of course, the application of bad bearings with much friction is not, what we recommend. Preferable is the well-controlled extraction of beneficial mechanical power from the rotating shaft. In the ideal case, we could have some active control, to influence the mechanical power extraction in such a way, that the "beneficial phase-shift" will always be regulated most close to its optimum value of 45°. This would allow long-term stable operation of the EMDR-motor with a maximum of extracted energy. This regulation can be constructed with regard to the angular velocity, or with regard to the phase shift. If we decide to choose the first alternative, we can define an "upper-level" and the "lower-level" for the angular velocity, and as soon as the rotating magnet becomes faster than the "upper-level", the amount of the power being extracted can be enhanced. Analogously, the amount of power being extracted can be reduced, as soon as the angular velocity becomes slower than the "lower-level". This is the basic idea on which the following chapter 6 is constructed.

6. The EMDR-Converter with mechanical power-extraction

As we saw in the figures 13, 14, 15 and especially in figure 19, the conversion of zero-point-energy in our EMDR-system can work efficiently during the initialisation of the operation. But the converted power goes asymptotically down to zero, in the same way as the phase shift between the rotation of the magnet and the LC-oscillation-circuit goes asymptotically down to zero, if we do not have some special technique maintain some "beneficial phase-shift" remarkably different from zero. We know the reason from section 5.

Therefore we have two apply mechanical torque $M_{mech}(t)$ to the shaft, in order to get a beneficial operation-mode of the EMDR-system. This torque $M_{mech}(t)$ has to be brought into equation (24a), as additional contribution, additionally to the torque of the magnetic forces (between permanent magnet and coil). The power which is extracted from the shaft by this torque is written in equation (27), and we find it in the second last line of table 1.

$$P_{mech} = M_{mech} \cdot \dot{\varphi} \qquad (27)$$

Mechanical power-extraction, due to the torque
$M_{mech} = c_r \cdot \dot{\varphi}$ with $c_r =$ coefficient of friction
with a torque proportional to the angular-velocity of the rotation.

We will see, that this will enable us to run the EMDR-motor long-term stable and extract remarkable power. The DFEM-simulation is very encouraging, and therefore it's source-code is printed completely in the appendix of the publication here.

What we apply is a torque proportional to the angular velocity of the rotation (this is an arbitrary decision, it could also be made different), which we find in the source-code in those lines which begin with the comment "{GG}". The coefficient of friction has the name "cr" in the program, and it can be controlled with the subroutine "Reibung_nachregeln". This subroutine works with an angular velocity called "phipZiel", around which we have a small hysteresis, within which the angular velocity of the EMDR-motor shall be kept. In order to make the results convincing (with regard to the conversion of ZPE-energy), we allow the magnet to spin a little bit faster than with the initial frequency, so that everybody can see that also the angular velocity is enhanced during operation, fed by ZPE-energy.

As can be seen in the input-data of the source-code, the solution of the differential equation is made by numerical iteration with 10^8 steps, of which everyone has 43 nanoseconds. It was verified, that this is indeed a sufficient time-resolution, so that the algorithm has converged to the serious result.

When you run the algorithm, you get the following data onto the screen:

```
power gained within the system:    5.04845399 Watt
total extraction of energy on the load resistor = 223.50737922 Joule
corresponding to a power of:       5.19784602848813E+0001 Watt
energy being supplied from input:  0.00000000000000E+0000 Joule
corresponding to a power of:       0.00000000000000E+0000 Watt
totally extraction of mechanical energy =   2271.25431928806 Joule
corresponding to a power of =               528.19867890420 Watt
at a duration of observation of    4.30000000000000E+0000 sec.
```

We interpret this as following:

The start of the EMDR-converter is initialised with an angular velocity of 30,000 rounds per minute, while there is no initial electrical energy within the LC-oscillation-circuit. The rotation of the magnet induces a voltage into the coil and thus brings electrical energy into the LC-oscillation-circuit. Of this reason, the initialisation of the operation extracts some energy from the rotation of the magnet and brings it into the LC-oscillation-circuit. We see this in figure 22. But even during the first oscillation we see the energy gain from the ZPE-energy, so that the angular velocity of the magnet will be soon faster than at the very beginning. There we see the transient behaviour of the system, during which the energy is distributed between the electrical and mechanical part of the system as the laws of physics require. The transient behavior finally leads us to the frequency "phipZiel" with precision of a small hysteresis due to the control mechanism of the "beneficial phase-shift".

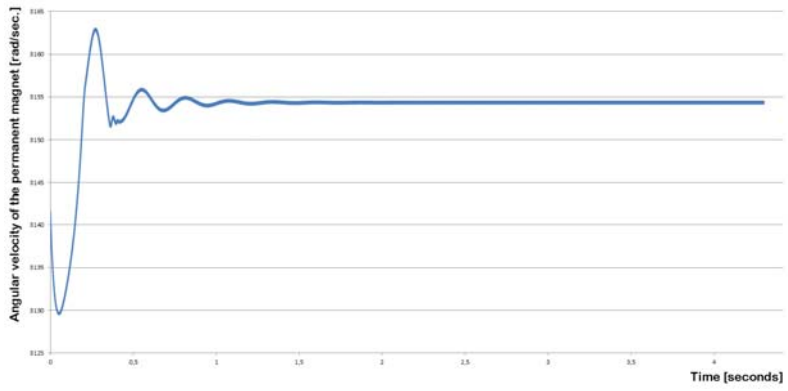


Fig.22:

Angular velocity of the rotating permanent magnet in a powerful EMDR-converter.

The graphic is plotted from column „I" of the Excel data-export.

The control of the power extraction is being done via a time-dependent regulation of the coefficient of friction c_r. The time-dependent dynamic behaviour of this coefficient is printed in figure 23. Obviously the transient motion at the very beginning of the operation requires some control of the coefficient. But is seems as if this control is not further necessary, when the stable mode of operation is achieved. (We will discuss this later in detail.)

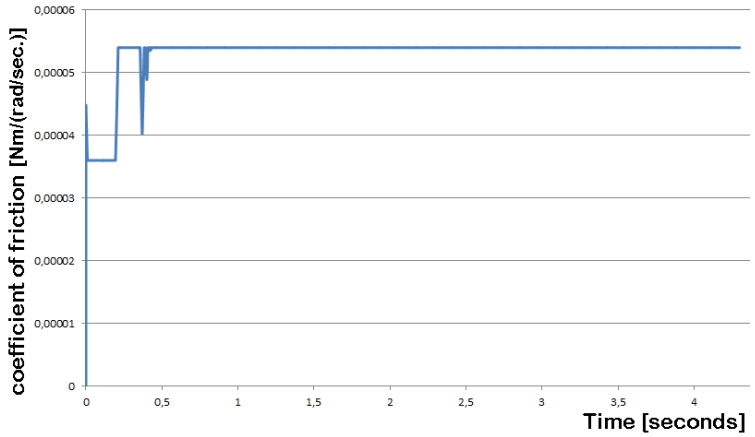


Fig.23:

Dynamic behaviour of the coefficient of friction, as it is used for the control of mechanical power extraction from the EMDR-motor.

The graphic is plotted from column „X“ of the Excel data-export.

The amount of power actually being extracted is plotted in figure 24. With regard to the size of the setup, a mechanical power-output of a bit more than 530 whites is okay - as the permanent magnet has a length of only 10 cm. And it shall be mentioned that there is additional electrical power-output and the machine (at the load resistor), which is not yet fully optimised with regard to power-output.

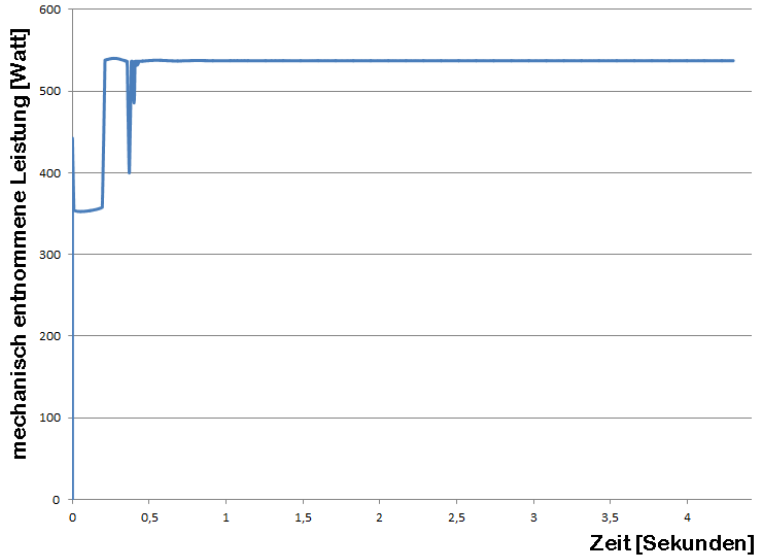


Fig.24:

Mechanical power output from the EMDR-converter.

The graphic is plotted from column „Y“ of the Excel data-export.

In order to get an imagination about the mechanical and electrical dimensions, necessary for the design of the prototype, we now want to inspect the electrical current and the voltage in the LC-oscillation-circuit (see figure 25).

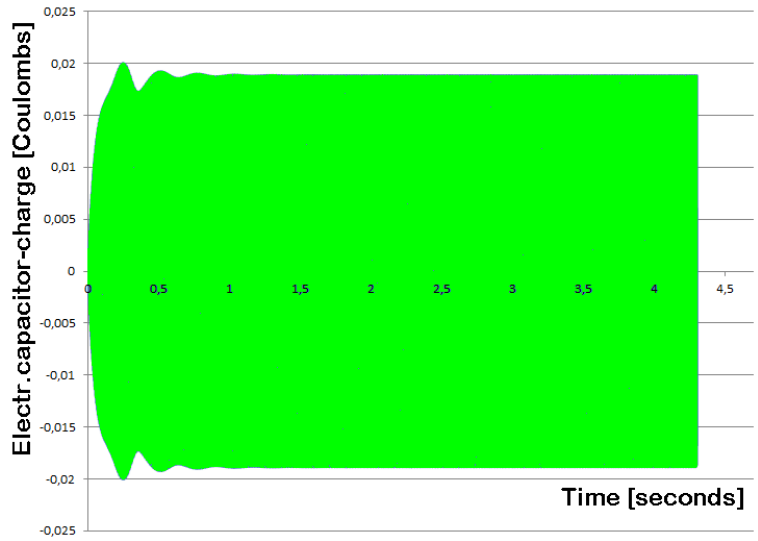


Fig.25:

Electrical charge Q in the capacitor of the EMDR-converter of our example.

The graphic is plotted from column „A“ of the Excel data-export.

The capacitor-voltage goes rather quick from the transient behaviour to the stable operation. Figure 25 displays the envelope of the oscillating charge in the capacitor, from which we calculate the capacitor-voltage according to equation (28), as being nearly 200 Volts. This value is not a problem for a practical experiment.

But it is important to know, that the capacitor must be tuned extremely fine, because this is the device, with which the EMDR-converter is fine-tuned. The adjustment of its value with a precision of about 1% … 0.1% is necessary, so that it is a good advice to use a capacitor bank for instance as shown in figure 26. All single capacitors must be connected parallel and not in series. The capacitor should have almost the same internal resistance. And it is recommendable that they have the same time constants for being charged and discharged. Nevertheless they shall have different capacity in order to make the fine-tuning possible over a wide range of capacity.

Finally it should be emphasized, that the adjustment of the capacitor is the means, by which the uncertainties and the approximations of the theoretical calculations have to be compensated !

This means, that the computation can deliver a theoretical value of the capacity deviating from the real experimental value even by a factor of 1 … 2 … 3 (or more ?).

$$U_C = \frac{Q}{C} = \frac{0.02\,C}{101.7\,\mu F} = 197V \tag{28}$$

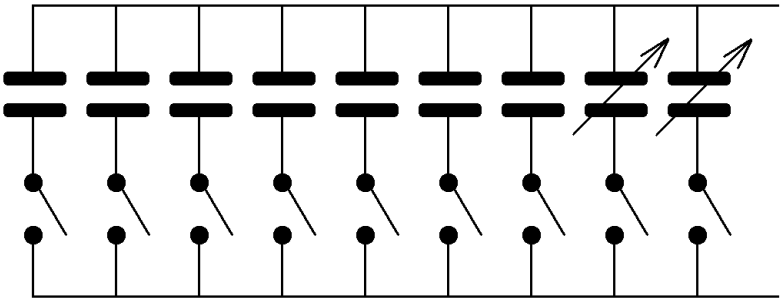


Fig.26:

Capacitor bank for fine-tuning of the capacity.

All single capacitors may have different capacity, so that a wide range of capacity can be tuned to this very fine adjustment.

An electrical current of about 60 Amperes (see figure 27) is not very much for the coil we have used, which has a diameter of the wire of 10 mm (i.e. a cross section 78.5 mm²). The problem is the handling of such a thick wire, but a trained mechanician should be able to do this. The capacitor bank must withstand a current of 60 Amperes.

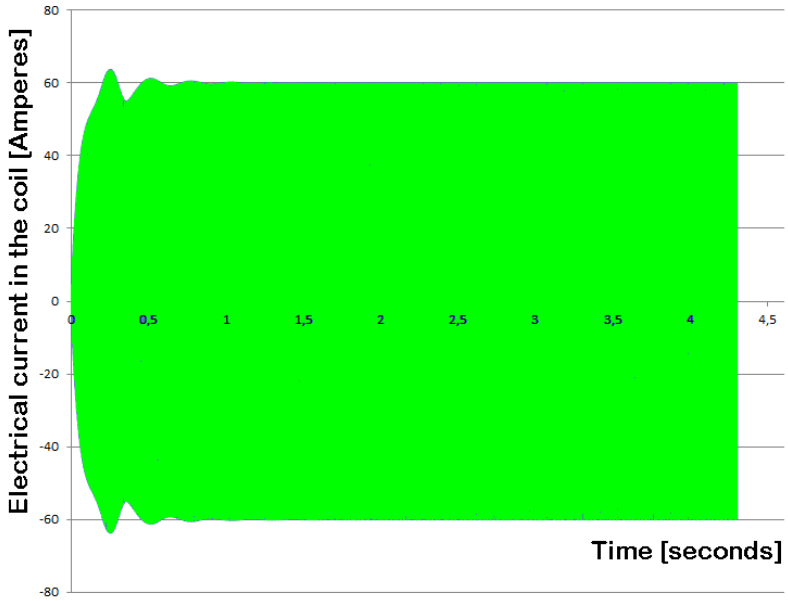


Fig.27:

Electrical current $\dot{Q}$ in the LC-oscillation-circuit of our EMDR-converter.

The graphic is plotted from column „B“ of the Excel data-export.

The voltage over the coil can be found by equation (29) from the law of induction. It's amplitude is also close to 200 Volts, and thus far away from causing difficulties.

$$U_L = -L \cdot \dot{I} = 9.936 \cdot 10^{-4}\, Henry \cdot 200000 \frac{A}{\sec} = -199\, Volt \qquad (29)$$

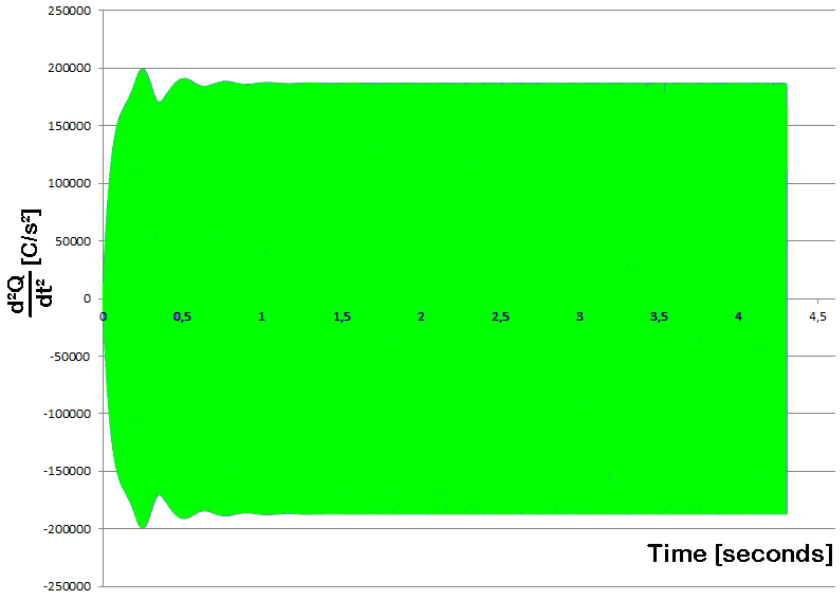


Fig.28:

Time derive of the current in the coil, i.e. $\ddot{Q}$ in our EMDR-converter.

The graphic is plotted from column „C" of the Excel data-export.

The angular acceleration of the rotating magnet (column "J" in Excel) with an amplitude of nearly 1000 rad/s^2 makes us understand, that the components of the EMDR-converter have to be mounted with appropriate mechanical stability. This angular acceleration acts on the magnet with a ponderable mass of 245 Gramms and a moment of inertia of rotation of $2.1 \cdot 10^{-4}$ kg·m^2. If the coil and the axis of rotation is not mounted with appropriate stability, the experiment might be dangerous.

The total energy of the system remains rather close to the value to which it is regulated (keep hysreresis in mind), as we see in figure 29. Similar behaviour can be seen in all channels of the energy analysis of the EMDR-system, because in the stable and durable long-term mode of operation, the energy is just oscillating between the different components of the system. And the amount of power per time, which is gained from the ZPE-energy is immediately converted into mechanical energy per time, being extracted at the shaft.

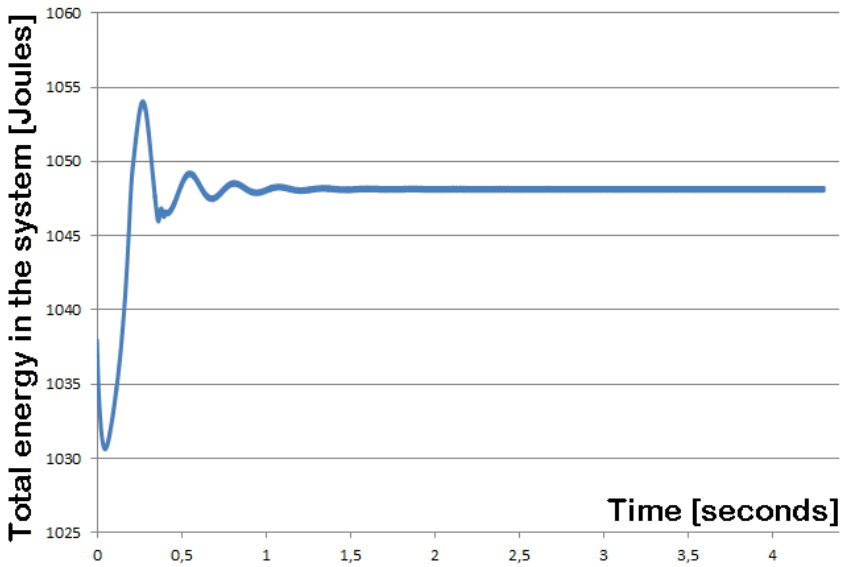


Fig.29:

Total energy in the EMDR-system.

The graphic is plotted from column „T" of the Excel data-export.

Additional question:

Is it allowed to drive the EMDR-motor without elaborated regulation of mechanical power extraction?

This would make it much easier to build up a prototype.

Answer:

Yes, it is allowed. It is even not the problem at all.

Also when the coefficient of friction is simply constant (in algorithm, const: „cr = crAnfang“), the EMDR-motor will run into a stable mode of operation by alone, as long as the coefficient of friction is within certain limits, which are not extremely narrow. If for instance we run the algorithm as shown in the appendix, but let us apply one single change, namely to use a constant coefficient of friction without any regulation, applying the value “crAnfang:=37E-6”, the motor will come rather quick to a stable mode of operation, as we can see it in figure 30. Also the mechanically extracted power will come to a constant value rather soon, as we see in figure 31.

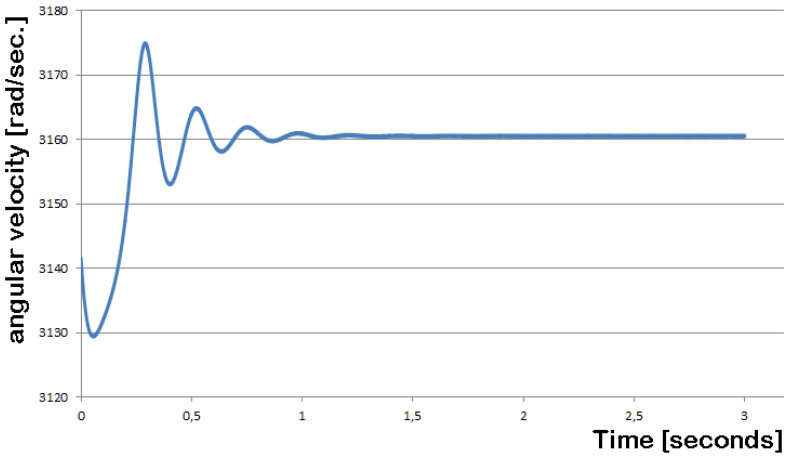


Fig.30:

Angular velocity of the rotating permanent magnet.

The graphic is plotted from column „I“ of the Excel data-export.

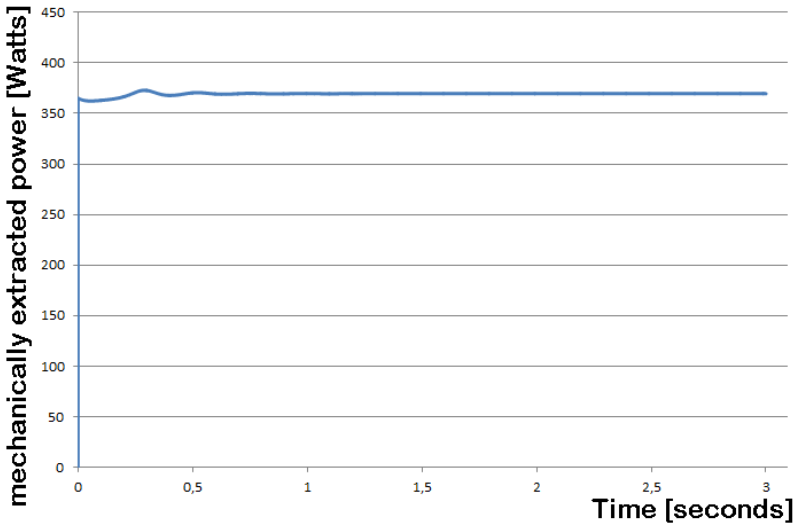


Fig.31:

Mechanical power extraction from the rotating shaft. The load-torque is proportional to the angular velocity of the axis.

The graphic is plotted from column „I“ of the Excel data-export.

Experimentalists, who build up an EMDR-converter with simple constant friction c_r, should be careful not to use to strong friction, because strong mechanical load during the transient (initialization-) phase of operation does extract too much energy, so that the system cannot come into the long-term stable mode of operation. The allowed mechanical load without regulation is smaller than the allowed mechanical load with regulation. This means, if we take the average value of the load-coefficient of figure 23 (this is cr:=crAnfang:=54E-6), the EMDR-converter would never come to a stable mode of operation, because it would be slowed down too much during the initial phase. If we apply “crAnfang:=54E-6” with good regulation of “cr”, we can extract 537 Watts mechanically, but if

we apply “cr:=54E-6=const” without any regulation, EMDR-converter does work as a self-running ZPE-engine.

Here are some examples for load-coefficients recommendable or not recommendable:

cr:=crAnfang:=$2.0\cdot10^{-5}$ => Pmech = 201 Watt (goes to stable operation)

cr:=crAnfang:=$2.5\cdot10^{-5}$ => Pmech = 251 Watt (goes to stable operation)

cr:=crAnfang:=$3.0\cdot10^{-5}$ => Pmech = 300 Watt (goes to stable operation)

cr:=crAnfang:=$3.5\cdot10^{-5}$ => Pmech = 349 Watt (goes to stable operation)

cr:=crAnfang:=$3.7\cdot10^{-5}$ => Pmech = 369 Watt (goes to stable operation)

cr:=crAnfang:=$4.0\cdot10^{-5}$ => Pmech = 247 Watt (no stable operation, angular velocity goes asymptotically down to zero)

cr:=crAnfang:=$5.3\cdot10^{-5}$ => Pmech = 117 Watt (average of Fig.23, no stable operation)

Obviously the extractable mechanical power increases linearly with increasing load-coefficient, as long as the mechanical load is not to strong (see the range of cr:=$2.0\cdot10^{-5}$ … $3.7\cdot10^{-5}$). But if the load is too strong, the EMDR-engine will not have a chance to adjust the “beneficial phase-shift” between its electrical and its mechanical motion in such a way, that it can find its stable operation. This is, what we observe at cr:=$4.0\cdot10^{-5}$, and of course much more at cr:=$5.3\cdot10^{-5}$.

So we can say, that the control of the load coefficient is not absolutely necessary, especially not for the first prototypes, but it is nice to have for EMDR-motors build later, in order to maximise the extractable power in technical application. This is, what we got from regulation of the load-coefficient:

initial phase: crAnfang:=$4.5\cdot10^{-5}$,

after regulation : cr:=$5.4\cdot10^{-5}$, => Pmech = 537 Watt (long-term operation, figure.23)

The reason is, that the load is regulated down in those moments in which the torque is weak (during initialization), and the load is regulated up during those moments in which strong torque and power is available.

7. Practical advice for experimenalists, who want to build an EMDR-Converter

With the end of section 6, the theory of the EMDR-system is discussed. But we now want to speak about practical aspects, which are interesting for those, who want to try to verify the theory experimentally [PC 11].

Central part of the EMDR-converter is the magnet. But the algorithm allows to simulate almost every available magnet or configuration of magnets. For the sake of simplicity, our calculation-example was done with the simple cylindrical bar-magnet, which should be available very easily. In [Tur 11], the calculation-example was restricted to a homogeneous magnetic field of a disc magnet, with an orientation of the magnetisation “in plain” (in order to make the computations easy). This restriction is now not necessary any further, with the new algorithm presented here. Thus the calculation-example in [Tur 11] has been a rather rough approximation, but the accuracy now is better.

Experimentalists who use a cylindrical bar-magnet should not forget about the aerodynamic drag of the rotating magnet due to its rather high angular velocity. We can accept this aerodynamic drag as a mechanical load which helps to maintain the “beneficial phase-shift” according to figure 20, when we will try to build the first prototypes.

But this is not the version which shall be used for future technical applications. For technical applications, engineers will have to try to maximise the output-power, and thus the aerodynamic drag should be minimized. This can be for instance done by inserting the cylindrical bar-magnet into a round disk, as shown in figure 32. But the disc must not be made of ferromagnetic material, because such a disc would guide the magnetic field flux lines into the totally wrong direction, or even keep the magnetic field flux lines inside the disc. And probably it is recommendable, to manufacture the round carrier-disc from nonconductive material in order to avoid eddy-currents carefully. Thus it is a good idea, to manufacture the round carrier-disc from plastic, and to use a type of plastic which is mechanically stable, so that it can withstand the centrifugal forces.

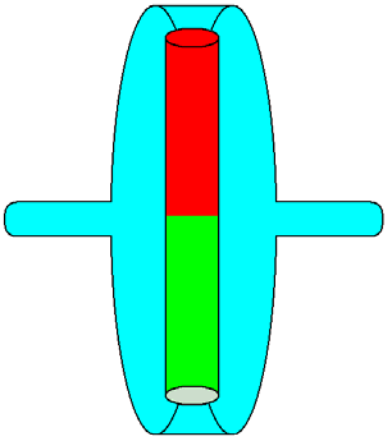


Fig.32:

Possible suggestion to encapsulate a cylindrical bar-magnet in order to minimise the aerodynamic drag due to the rotation of the magnet.

Regarding eddy-currents: Up to now, there are no investigations about the question, whether they disturb the EMDR-motor or not. There is even no theoretical analysis of the role of eddy-currents up to now. There have been hints in discussions with colleagues, that eddy-currents would prevent the EMDR-motor to run at all, but perhaps eddy-currents might perhaps define a mechanical load and thus help to maintain the "beneficial phase-shift". It should be part of the experimental investigations to find out, whether eddy-currents disturb the EMDR-system or not. But for the first experimental trials and approaches, I recommend to avoid eddy-currents, because the theory works fine without eddy-currents.

Several colleagues discussed about the question, whether rare-earth magnets (such as neodymium) are mechanically stable enough to withstand the centrifugal-forces at 30,000 rounds per minute. It is well-known that neodymium magnets are less mechanically stable than steel. Nevertheless an encapsulation with steel in order to enhance the mechanical stability is absolutely forbidden, due to its influence on the field flux lines. For those, who want to maximise the mechanical stability, it is recommended to use an Iron-Cobalt-Nickel alloy for the magnet (without any encapsulation). It is possible to make a cylindrical bar-magnets of such alloy with a magnetic field-strength of 1 Tesla at each end of the bar. Such field-strength is absolutely sufficient for the operation in the EMDR-system. And furthermore the application of magnets of such alloy allows the mechanical stability of steel without encapsulating the magnet at all.

For sure the problem of eddy-currents is not neglectable for any encapsulation. We can see this, when we regard the energy-transfer from the rotating magnet into the coil as some special type of eddy-current-loss, which we need for the operation of the EMDR-converter.

Nevertheless, eddy-current-losses in the axis of rotation (we speak about the axis necessary to keep the magnet in its rotating-position), should not be regarded as a very large problem. It was discussed to use special glassfibre-plastic (for instance "GFK") in order to get an axis without any eddy-current-losses. There is no argument against such an axis, as long as it is stable enough. But due to the stability of such a material, it might be recommendable, to mount the bearings rather close to the magnet (as drawn in figure 11), which makes it difficult, to apply the initial angular velocity as well as

to extract mechanical power during operation. Furthermore, if eddy-current-losses are the reason to use an axis made of glassfibre-plastic, the bearings, which are rather close to the magnet and inside the coil, should also be of some isolating material, as for instance ceramic-ball-bearings. But such bearings are not stable for the same angular frequency as steel bearings. This is the reason, why I do not want to exclude the use of non-ferromagnetic metallic axis, which is long enough, to mount the steel-bearings far away from the magnet and the coil (i.e. outside the coil). Finally the experiment will have to decide between these different suggestions for the setup - some of them might work and some others not.

Also a second coil has been under discussion, which could be brought to the position in which we have the input-coil, but which has the purpose is to extract energy, namely electrically induced energy due to the rotation of the magnet. This application would change the name of the “input-coil” into “output-coil”. But the algorithm in the appendix makes it rather easy to add such an output-coil into our EMDR-model. Such an output-coil would have the advantage, that we can adjust the number of windings to the requirements of the voltage-current-characteristics, which will be preferable for future applications. (The output-coil can also be mounted perpendicular to the turbo-coil in order to allow both of them to come most close to the rotating magnet.)

A further suggestion regards the reduction of the angular velocity of the rotation of the magnet, namely the use of a multipole-magnet (of higher order), as it can be seen for instance in figure 33. There we have 16 bar-magnets (dipoles) mounted around a wheel, so that the whole magnetic setup has a multi-polarity of 16, instead of 2, as we have it with simple cylindrical bar-magnet. The consequence is, that such a multipole-wheel will have 16 changes of the magnetic polarity within every turn instead of 2. An enhancement of the number of polarity-changes per turn by a factor of 8 means, that we can reduce the angular velocity (of the magnet) by a factor of 8 (in comparison with the simple cylindrical bar-magnet). A simple numerical example shows the advantage: If we for instance have an electrical LC-oscillation-circuit with a resonance frequency of the 32,000 rounds per minute, the mechanical rotation of the multipole-wheel only requests 4000 rounds per minute. The numerical example can be changed arbitrarily, as long as the number of the multipoles is even (not odd).

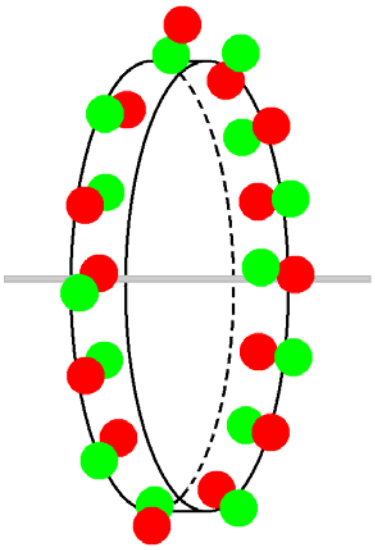


Abb.33:

Multipole-Magnet, which is manufactured by mounting several magnetic dipoles around a wheel.

Nevertheless it should be pointed out again, that a dipole-magnet has to be mounted mechanically very stable, same as a multipole-magnet of higher-order in order to avoid accidents. But the situation is less critical for multipole-magnet of higher order, because it runs more smoothly than dipole magnet (similar as every electrical engine).

8. Resumée

The result of the work presented here, is a computation method for ZPE-motors, which is much more close to reality than the very first development in [Tur 11], which explained the very principle more than the engineering aspects. The algorithm in the appendix is written in a way, that every trained researcher should be able to use it for the simulation of his own ZPE-converter, which can be the EMDR-design (invented by the author of this article) or which can also be some other design.

Important part of the work presented here, is the design of concrete example for a self-running ZPE-motor (this is the EMDR-engine). The design is calculated and suggested practicable enough, that every trained experimentalist should be able to build it up, as soon as he or she has the possibility to work in an appropriately equipped laboratory. Of course there are still several open experimental questions to be solved, but I am convinced, that this should not be an existential problem. (It might be a problem of time, and thus I do not dare any predictions.)

The author of this article would like to build up such a prototype by himself very much (especially on the background that he is educated/trained as an experimental physicist), but unfortunately he does not have any possibility to work in a laboratory in the moment now.

9. Literature References

[Ans 08] Finite Element Program ANSYS, John Swanson (1970-2008)
ANSYS, Inc. Software Products, http://www.ansys.com

[Bea 02] Motionless Electromagnetic Generator, Tom Bearden et. al.
US Patent, 6,362,718 vom 26. März 2002
Inventor: Patrick L. Stephen, Thomas E. Bearden, James C. Hayes, Kenneth D. Moore, James L. Kenny, To be found also at http://www.cheniere.org

[Bor 99] Borland Pascal (Delphi 5 from 1999 or newer version)

[Bro 08] Taschenbuch der Mathematik
Ilja N Bronstein, Konstantin A Semendjajew, Gerhard Musiol, Heiner Mühlig,
Verlag Harri Deutsch, 7.Auflage, 2008, ISBN 978-3-8171-2017-8

[Dub 90] Dubbel - Taschenbuch für den Maschinenbau , 17.Auflage
W. Beitz, K.-H. Küttner et. al., Springer-Verlag. 1990. ISBN 3-540-52381-2

[Ger 95] Gerthsen Physik, H. Vogel
Springer Verlag. 1995. ISBN 3-540-59278-4

[Hoh 11] http://www.magnetmotor.at/ by Dietmar Hohl
In Internet many self-running magnet-motors can be found. Here is one example.

[Hur 40] The Invention of Hans Coler, Relating To An Alleged New Source Of Power.
R. Hurst, B.I.O.S. Final Report No. 1043, B.I.O.S.Trip No. 2394
B.I.O.S. Target Number: C31/4799, British Intelligence Objectives Sub-Committee

[Jac 81] Klassische Elektrodynamik, John David Jackson
Walter de Gruyter Verlag. 1981. ISBN 3-11-007415-X

[Jeb 06] Die Urkraft aus dem Universum, Klaus Jebens
Jupiter-Verlag. 2006. ISBN 3-906571-23-8

[Kep 10] Keppe Motor Manual 1.0, 2008, Norberto Keppe and Cláudia B. S. Pacheco.
http://www.keppemotor.com/ and http://www.keppemotor.com/manual1.php

[Koh 96] Praktische Physik, 3 Bde., Friedrich Kohlrausch, et. al.
Verlag B.G.Teubner Stuttgart, 1996, ISBN-13: 978-3519230014

[Mar 88-98]
- Der Kugellager-Motor und der Huber-Effekt,
  Stefan Marinov, raum&zeit 32 (1988) 81-84
- Repetition of Silvertooth's experiment for measuring the aether drift",
  Stefan Marinov, Speculations in Science and Technology 12 No.3 (1989) 187-179
- Propulsive and Rotating Ampère Bridges and the Principle of Relativity
  Stefan Marinov, Physics Essays 4 No.1 (1991) 30-36
- The Missing Magnetic Force Law, Galilean Electrodynamics 9 No.2
  Stefan Marinov, (March/April 1998) 35-37

[Mie 84] Kompendium Hypertechnik. Tachyonenenergie, Hyperenergie, Antigravitation.
Sven Mielordt, Berlin, 1984
Reprint, 4. Auflage, raum&zeit Verlag, ISBN 3-89005-005-0

[Nie 83] Konversion von Schwerkraft-Feld-Energie. Revolution in Technik, Medizin, Gesellschaft.
Hans A. Nieper
MIT-Verlag, Oldenburg, 1983, 4. erw. Auflage, ISBN 3-925188-00-2

[PC 11] private communication
with numerous colleagues – too many people to be enumerated here.

[Stö 07] Taschenbuch der Physik, Horst Stöcker
Verlag Harri Deutsch. 2007. ISBN-13 987-3-8171-1720-8

[Tur 10a] Example of a simple Algorithm for the Construction of Zero-point-energy Converters
Claus W. Turtur, PHILICA.COM, ISSN 1751-3030, Article number 207, (9. Okt. 2010)

[Tur 10b] DFEM-Computation of a Zero-point-energy Converter with realistic Parameters for a practical setup
Claus W. Turtur, PHILICA.COM, ISSN 1751-3030, Article number 213, (7. Dez. 2010)

[Tur 11] DFEM-Simulation of a Zero-point-energy Converter with realisable Dimensions and a Power-output in the Kilowatt-range.
Claus W. Turtur, PHILICA.COM, ISSN 1751-3030, Article number 219, (7. Feb. 2011)

10. Appendix: Source-Code of the DFEM-Algorithm (EMDR_009i.dpr) [Bor 99]

Hint: Those lines of the Pascal-Program, which are longer than the text lines of the publication are continued right-aligned. (If you want to run the program, you should remove this alignment, or easier use the source-code to be downloaded for free from my internet-page.)

```
Program KM_009i;
{$APPTYPE CONSOLE}
uses
  Windows,
  Messages,
  SysUtils,
  Classes,
  Graphics,
  Controls,
  Forms,
  Dialogs;

Const Bn=7;          {number of steps for the data-storage of the magnetic field-strength}
Const SpNmax=200;    {maximal number of possible support points for the meshing of the coils}
Const FlNmax=2000;   {maximal number of possible finite area-elements of the coils}
Const MESEanz=200;   {actual number of elements for the permanent-magnet emulation-coils}
Const AnzPmax=35000; {Dimension of the Arrays for data-export to Excel)}

Var epo,muo  : Double;  {constants of nature}
    Bsw      : Double;  {step width for the storage of the magnetic field}
    Spsw     : Double;  {step width for the mesh-generation of the coils}
    SpN      : Integer; {number of support points of the coils}
    FlN      : Integer; {number of area elements for mesh generation of the coils}
    LiGe     : Double;  {speed of light}
    xo,yo,zo : Integer; {Geometrical parameters Fig.1}
    Ninput   : Integer; {number of windings of the Input-coil}
    Nturbo   : Integer; {number of windings of the Turbo-coil}
    PsiSFE   : Double;  {magnetic flux through every area element of the coils}
    PsiGES   : Double;  {magnetic flux through the total coil}
    B1,B2,B3,B4,B5      : Double;  {Fourier-coefficients, general}
    B1T,B2T,B3T,B4T,B5T : Double;  {Fourier-coefficients, Turbo-coil}
    B1I,B2I,B3I,B4I,B5I : Double;  {Fourier-coefficients, Input-coil}
    B1dreh,phase : Double;  {coefficients for fast torque computation}
    MEyo, MEro, MEI : Double;{dimensions of the coils for the emulation of the permanent magnet}
    Bx,By,Bz : Array [-Bn..Bn,-Bn..Bn,-Bn..Bn] of Double; {Cartesian components from magnetic induction}
    MESEx,MESEy,MESEz    : Array [1..MESEanz] of Double;{position of the magnet emulation coils}
    MESEdx,MESEdy,MESEdz : Array [1..MESEanz] of Double;{current direction within magnet emulation coils}
    OrtBx,OrtBy,OrtBz : Array [-Bn..Bn,-Bn..Bn,-Bn..Bn] of Double; {Cartesian components for determination
                                                                    of the magnetic field}
    SpIx,SpIy,SpIz    : Array [1..SpNmax] of Double; {support points for polygonial line, Input-coil}
    SpTx,SpTy,SpTz    : Array [1..SpNmax] of Double; {support points for polygonial line, Turbo-coil}
    SIx,SIy,SIz       : Array [1..SpNmax] of Double; {centre of conductor loop elements, Input-coil}
    STx,STy,STz       : Array [1..SpNmax] of Double; {centre of conductor loop elements, Turbo-coil}
    dSIx,dSIy,dSIz    : Array [1..SpNmax] of Double; {direction vector,conductor loop elements,Input-coil}
    dSTx,dSTy,dSTz    : Array [1..SpNmax] of Double; {direction vector,conductor loop elements,Turbo-coil}
    FlIx,FlIy,FlIz    : Array [1..FlNmax] of Double; {area elements, Input-coil, Cartesian coordinates}
    FlTx,FlTy,FlTz    : Array [1..FlNmax] of Double; {area elements, Turbo-coil, Cartesian coordinates}
    BxDR,ByDR,BzDR          : Array [-Bn..Bn,-Bn..Bn,-Bn..Bn] of Double; {rotated magnetic field}
    OrtBxDR,OrtByDr,OrtBzDR : Array [-Bn..Bn,-Bn..Bn,-Bn..Bn] of Double; {rotated position vectors}

    {Zum Lösen der Bewegungs-Differentialgleichung:}
    phi,phip,phipp      : Array[0..AnzPmax] of Double; {orientation angle of the magnet, derivatives}
    Q,Qp,Qpp            : Array[0..AnzPmax] of Double; {electrical charge in Turbo-coil}
    QI,QpI,QppI         : Array[0..AnzPmax] of Double; {electrical charge in Input-coil}
    phio,phipo,phippo,phim,phipm,phippm : Double; {angle and derivatives}
    qoT,qpoT,qppoT,qmT,qpmT,qppmT       : Double; {electrical charge in Turbo-coil and derivatives}
    qoI,qpoI,qppoI,qmI,qpmI,qppmI       : Double; {electrical charge in Input-coil and derivatives}
    PSIinput,PSIturbo   : Array[0..AnzPmax] of Double; {magnetic flux in the coils: Input & Turbo}
    UindInput,UindTurbo : Array[0..AnzPmax] of Double; {induced voltage, Input- and Turbo- coil}
    UinduzT,UinduzI     : Double;                      {induced voltage in the moment "NOW"}
    i                   : LongInt;  {control-variable to count time-steps}
    AnzP,AnzPmerk       : LongInt;  {total number of times steps for the solution of DGL.}
    dt                  : Double;   {duration of times steps}
    PlotAnfang,PlotEnde,PlotStep : LongInt; {control variables for data-export to Excel}
    Abstd               : Integer;  {plot control during initialisation}
    znr                 : Integer;  {plot control for data-export to Excel}
    LPP                 : Integer;  {last plot-point, to be used for data-export}
    Zeit : Array[0..AnzPmax] of Double; {time-scale}
    KG,KH,KI,KJ,KK,KL,KM,KN,KO,KP : Array[0..AnzPmax] of Double; {arrays for data-export to Excel}
    KQ,KR,KS,KT,KU,KV,KW,KX,KY,KZ : Array[0..AnzPmax] of Double; {arrays for data-export to Excel}
```

```
    BTx,BTy,BTz : Double;  {magnetic field of the Turbo coil at any position}
    BIx,BIy,BIz : Double;  { magnetic field of the input coil at any position }
    merk        : Double;  {for the purpose of testing}
    schonda     : Boolean; {have these data already be initialized ?}
    DD,DLI,DLT  : Double;  {diameter and length of the wires of the coils}
    rho         : Double;  {Ohm*m}  {specific resistance of copper, Kohlrausch,T193}
    RI,RT       : Double;  {Ohm's resistance of the coils}
    CT          : Double;  {Farad}  {capacitor in the Turbo-circuit}
    CI          : Double;  {Farad}  {capacitor in the Input-circuit}
    LT,LI       : Double;  {intductivity of the coils, Turbo and Input}
    nebeninput  : Double;  {windings side-by-side, Input-coil}
    ueberinput  : Double;  {windings on top of each other, Input-coil}
    nebenturbo  : Double;  {windings side-by-side, Turbo-coil}
    ueberturbo  : Double;  {windings on top of each other, Turbo-coil}
    BreiteI,HoeheI,BreiteT,HoeheT : Double; {wide and height of the coil bodies}
    omT,TT      : Double;  {angular frequency of Turbo oscillation circuit LT & CT.}
    UmAn,omAn   : Double;  {initial angular velocity of the magnet}
    UmSec       : Double;  {initial angular velocity of the magnet}
    J           : Double;  {moment of inertia of rotation of the magnet}
    rhoMag      : Double;  {density of the magnet-material}
    Mmag        : Double;  {Mass of the magnet}
    Rlast       : Double;  {Ohm's load resistor in the Turbo circuit}
    Uc,Il       : Double;  {initial conditions electrically: capacitor-voltage, coil-current}
    Tjetzt      : Double;  {moment now for the solution of the differential equation}
    QTmax,QImax,QpTmax,QpImax,QppTmax,QppImax,phipomax : Double; {maximum values for screen display}
    Wentnommen : Double; {total extracted energy}
    AnfEnergie,EndEnergie : Double; {energy comparison within the system}
    steigtM,steigtO : Boolean;  {check the slope of the reference signal}
    Iumk            : LongInt;  {reverse point of the reference Input-Signal}
    fkI,fkT         : Double; {correction of interactivity}
    Pzuf,Ezuf       : Double; {supported power by Input-voltage}
    crAnfang,cr     : Double; {coefficient of friction proportional to angular velocity}
    phipZiel        : Double; {angular velocity for friction-control}
    Preib           : Double; {extracted mechanical power}
    Ereib           : Double; {extracted mechanical energy}

Procedure Dokumentation_des_Ergebnisses;
Var fout   : Text;
begin
  Assign(fout,'Auswertung'); Rewrite(fout); {open file}
  Writeln(fout,'DFEM-Simulation of a EMDR-Motor.');
  Writeln(fout,' ');
  Writeln(fout,'Parameters for the solution of the differential equation:');
  Writeln(fout,'AnzP = ',AnzP:12,' number of times steps in calculation');
  Writeln(fout,'dt   = ',dt:12,' {Seconds} duration of times steps for iteration.');
  Writeln(fout,'Abstd= ',Abstd:5,'  {only for preparation-work, do not alter the value}');
  Writeln(fout,'PlotAnfang = ',Round(PlotAnfang):10,' {first plot point, data-Export to Excel}');
  Writeln(fout,'PlotEnde   = ',Round(PlotEnde):10,' {last plot point, data-Export to Excel}');
  Writeln(fout,'PlotStep   = ',Round(PlotStep):10,' {step-width for data-Export to Excel}');
  Writeln(fout,' ');
  Writeln(fout,'input data for the coils:');
  Writeln(fout,'Automatic mesh generation.');
  Writeln(fout,'Spsw   = ',Spsw:12:6,' Meters: Coil-meshing in steps of Spsw');
  Writeln(fout,'xo = ',xo,',  number of steps of "Spsw" ');
  Writeln(fout,'yo = ',yo,',  number of steps of "Spsw" ');
  Writeln(fout,'zo = ',zo,',  number of steps of "Spsw" ');
  Writeln(fout,'Ninput = ',Ninput:9,'  number of windings Input- Coil');
  Writeln(fout,'Nturbo = ',Nturbo:9,'  number of windings Turbo- Coil ');
  Writeln(fout,'nebeninput = ',Round(nebeninput):9,' windings side-by-side Input- Coil');
  Writeln(fout,'ueberinput = ',Round(ueberinput):9,' windings on top of each other Input- Coil');
  Writeln(fout,'nebenturbo = ',Round(nebenturbo):9,' windings side-by-side Turbo- Coil');
  Writeln(fout,'ueberturbo = ',Round(ueberturbo):9,' windings on top of each other Turbo-coil');
  Writeln(fout,' ');
  Writeln(fout,'Bsw = ',Bsw:9,' store magnetic field in centimeter-steps');
  Writeln(fout,'emulation of the one Tesla magnet:');
  Writeln(fout,'MEyo = ',MEyo:14,'  y-coordinates of the magnet emulation coils');
  Writeln(fout,'MEro = ',MEro:14,'  Radius of the magnet emulation coils');
  Writeln(fout,'MEI  = ',MEI:14,'  Current in the magnet emulation coils');
  Writeln(fout,' ');
  Writeln(fout,'several technical values:');
  Writeln(fout,'DD = ',DD:12:7,'  {Meter} {diameter of the coil wire');
  Writeln(fout,'rho = ',rho,'  {Ohm*m}    {Spezific resistance of copper');
  Writeln(fout,'rhoMag = ',rhoMag,'  {kg/m^3} {Density of the magnet material');
  Writeln(fout,'CT = ',CT:14,'  {Farad}    {Turbo-Capacitor');
  Writeln(fout,'CI = ',CI:14,'  {Farad}    {Input- Capacitor');
  Writeln(fout,' ');
  Writeln(fout,'other values:');
  Writeln(fout,'Rlast = ',Rlast:15,'  {Ohm} load resistor in Turbo-circuit');
```

```
  Writeln(fout,'UmAn = ',UmAn:10:2,'  {U/min} mechanical initial conditions- Rotating Magnet');
  Writeln(fout,'Uc = ',Uc:10:2,'  {Volt} electrical initial conditions - voltage TURBO-capacitor');
  Writeln(fout,'Il = ',Il:10:2,'  {Ampere} electrical initial conditions - TURBO-current');
  Writeln(fout,' ');
  Writeln(fout,'Mechanical power-extraction : ');
  Writeln(fout,'Coeffizient of power-extraction: ',crAnfang:17:12,' Nm/(rad/s)');
  Writeln(fout,'angular velocity for control: ',phipZiel:17:12,' U/min');
  Writeln(fout,' ');
  Writeln(fout,'Other Parameters, no input');
  Writeln(fout,'DLI:=4*(yo+zo)*Spsw*Ninput = ',DLI:10:5,' {Meter} Length of coil-wire, Input-coil');
  Writeln(fout,'DLT:=4*(yo+zo)*Spsw*Nturbo = ',DLT:10:5,' {Meter} Length of coil wire, Turbo-coil');
  Writeln(fout,'RI:=rho*(DLI)/(pi/4*DD*DD) = ',RI:10:5,' {Ohm} Ohm`s resistance of Input-Coil');
  Writeln(fout,'RT:=rho*(DLT)/(pi/4*DD*DD) = ',RT:10:5,' {Ohm} Ohm`s resistance of Turbo-Coil');
  Writeln(fout,'BreiteI:=nebeninput*DD = ',BreiteI:10:5,' Width and Height of Input-Coil');
  Writeln(fout,'HoeheI:=ueberinput*DD  = ',HoeheI:10:5,' Width and Height of Input-Coil');
  Writeln(fout,'BreiteT:=nebenturbo*DD = ',BreiteT:10:5,' Width and Height of Turbo-Coil');
  Writeln(fout,'HoeheT:=ueberturbo*DD  = ',HoeheT:10:5,' Width and Height of Turbo-Coil');
  Writeln(fout,'fkI:=Sqrt(HoeheI*HoeheI+4/pi*2*yo*2*zo)/HoeheI = ',fkI:10:5,' correction of Induktivity
                                                                                  short Input-Coil');
  Writeln(fout,'fkT:=Sqrt(HoeheT*HoeheT+4/pi*2*yo*2*zo)/HoeheT = ',fkT:10:5,' correction of Induktivity
                                                                                  short Turbo-Coil');
  Writeln(fout,'LI:=muo*(2*yo+BreiteI)*(2*zo+BreiteI)*Ninput*Ninput/(HoeheI*fkI) = ',LI,' Induktivität
                                                                                       Input-Coil');
  Writeln(fout,'LT:=muo*(2*yo+BreiteT)*(2*zo+Breitet)*Nturbo*Nturbo/(HoeheT*fkT) = ',LT,' Induktivität
                                                                                       Turbo-Coil');
  Writeln(fout,'omT:=1/Sqrt(LT*CT) = ',omT,' oscillation frequency of Turbo circuit');
  Writeln(fout,'TT:=2*pi/omT = ',TT,' Period of Turbo-circuit LT & CT.');
  Writeln(fout,'Mmag:=rhoMag*(pi*MEro*MEro)*(2*MEyo) = ',Mmag:8:3,' kg Mass of the magnet');
  Writeln(fout,'J:=Mmag/4*(MEro*MEro+4*MEyo*MEyo/3) = ',J,' moment of inertia, Rotation of magnet');
  Writeln(fout,' ');
  Writeln(fout,'display of several Parameters:');
  Writeln(fout,'Magnet: Start-angular velocity.: omAn = ',omAn:15:6,' rad/sec');
  Writeln(fout,'Magnet: Start-angular velocity, Umdr./sec.: UmSec = ',UmSec:15:10,' Hz');
  Writeln(fout,'Mass of the Magnet = ',Mmag:10:6,' kg');
  Writeln(fout,'Tmoment of inertia, Rotation of magnet',J,' kg*m^2');
  Writeln(fout,'duration of observation: ',AnzP*dt,' sec.');
  Writeln(fout,'Excel-Export: ',PlotAnfang*dt:14,'...',PlotEnde*dt:14,' sec., Step ',PlotStep*dt:14,'
                                                                                          sec.');
  Writeln(fout,'These are ',(PlotEnde-PlotAnfang)/PlotStep:8:0,' Data-points (lines).');
  Writeln(fout,' ');
  Writeln(fout,' ********************************************************************************');
  Writeln(fout,' ');
  Writeln(fout,'some results of the computation:');
  Writeln(fout,'initial energy in the system:    ',AnfEnergie:14:8,' Joule');
  Writeln(fout,'final energy in the system:        ',EndEnergie:14:8,' Joule');
  Writeln(fout,'corresponding to a power of:',(EndEnergie-AnfEnergie)/(AnzP*dt):14:8,' Watt');
  Writeln(fout,'extracted to energy at load resistor = ',Wentnommen:14:8,' Joule');
  Writeln(fout,'ecorresponding to a power of:',Wentnommen/(AnzP*dt),' Watt');
  Writeln(fout,'input-energy: ',Ezuf,' Joule');
  Writeln(fout,'corresponding to a power of:',Ezuf/(AnzP*dt),' Watt');
  Writeln(fout,'totally extracted mechanical energy = ',Ereib:18:11,' Joule');
  Writeln(fout,'corresponding to a power of = ',Ereib/(AnzP*dt):18:11,' Watt');
  Writeln(fout,'duration of observation',(AnzP*dt),' sec.');
  Close(fout);
end;

Procedure Wait;
Var Ki : Char;
begin
  Write('<W>'); Read(Ki); Write(Ki);
  If Ki='e' then Halt;
  If Ki='E' then Halt;
  If Ki='d' then Dokumentation_des_Ergebnisses;
  If Ki='D' then Dokumentation_des_Ergebnisses;
end;

Procedure ExcelAusgabe(Name:String;Spalten:Integer);
Var fout   : Text;
    lv,j,k : Integer; {control variables}
    Zahl   : String;  {print values to excel}
begin
  Assign(fout,Name); Rewrite(fout); {File open}
  For lv:=0 to AnzP do  {from "plotanf" to "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=1 to Spalten do
      begin   {print columns, 3*charge, 3*angle, then 8 auxiliary}
```

```
        If j=1  then Str(Q[lv]:19:14,Zahl);
        If j=2  then Str(Qp[lv]:19:14,Zahl);
        If j=3  then Str(Qpp[lv]:19:14,Zahl);
        If j=4  then Str(phi[lv]:19:14,Zahl);
        If j=5  then Str(phip[lv]:19:14,Zahl);
        If j=6  then Str(phipp[lv]:19:14,Zahl);
        If j=7  then Str(KG[lv]:19:14,Zahl);
        If j=8  then Str(KH[lv]:19:14,Zahl);
        If j=9  then Str(KI[lv]:19:14,Zahl);
        If j=10 then Str(KJ[lv]:19:14,Zahl);
        If j=11 then Str(KK[lv]:19:14,Zahl);
        If j=12 then Str(KL[lv]:19:14,Zahl);
        If j=13 then Str(KM[lv]:19:14,Zahl);
        If j=14 then Str(KN[lv]:19:14,Zahl);
        For k:=1 to Length(Zahl) do
        begin   {use "commata" instead of decimal points}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Data-separation Tabulator}
      end;
      Writeln(fout,'');    {line-separation}
    end;
  end;
  Close(fout);
end;

Procedure ExcelLangAusgabe(Name:String;Spalten:Integer);
Var fout   : Text;    {timescale and up to 25 columns}
    lv,j,k : Integer; {control variables}
    Zahl   : String;  {print data to excel}
begin
  If (Spalten>25) then
  begin
    Writeln('FEHLER: Zu viele Spalten. Soviele Daten-Arrays sind nicht vorhanden.');
    Writeln(' => PROGRAMM WURDE ANGEHALTEN : STOP !');
    Wait; Wait; Halt;
  end;
  Assign(fout,Name); Rewrite(fout); {File open}
  For lv:=0 to LPP do  {from "plotanf" to "plotend"}
  begin
    If (lv mod Abstd)=0 then
    begin
      For j:=0 to Spalten do
      begin   {print columns, 3*charge, 3*angle, then auxiliary}
        If j=0  then Str(Zeit[lv]:19:14,Zahl);  {Markieren der Zeit-Skala}
        If j=1  then Str(Q[lv]:19:14,Zahl);     {Turbo-coil}
        If j=2  then Str(Qp[lv]:19:14,Zahl);    {Turbo-}
        If j=3  then Str(Qpp[lv]:19:14,Zahl);   {Turbo-coil}
        If j=4  then Str(QI[lv]:19:14,Zahl);    {Input-coil}
        If j=5  then Str(QpI[lv]:19:14,Zahl);   {Input-coil}
        If j=6  then Str(QppI[lv]:19:14,Zahl);  {Input-coil}
        If j=7  then Str(phi[lv]:19:14,Zahl);   {Magnet}
        If j=8  then Str(phip[lv]:19:14,Zahl);  {Magnet}
        If j=9  then Str(phipp[lv]:19:14,Zahl); {Magnet}
        If j=10 then Str(KK[lv]:19:14,Zahl);    {Auxiliary}
        If j=11 then Str(KL[lv]:19:14,Zahl);    {Auxiliary}
        If j=12 then Str(KM[lv]:19:14,Zahl);    {Auxiliary}
        If j=13 then Str(KN[lv]:19:14,Zahl);    {Auxiliary}
        If j=14 then Str(KO[lv]:19:14,Zahl);    {Auxiliary}
        If j=15 then Str(KP[lv]:19:14,Zahl);    {Auxiliary}
        If j=16 then Str(KQ[lv]:19:14,Zahl);    {Auxiliary}
        If j=17 then Str(KR[lv]:19:14,Zahl);    {Auxiliary}
        If j=18 then Str(KS[lv]:19:14,Zahl);    {Auxiliary}
        If j=19 then Str(KT[lv]:19:14,Zahl);    {Auxiliary}
        If j=20 then Str(KU[lv]:19:14,Zahl);    {Auxiliary}
        If j=21 then Str(KV[lv]:19:14,Zahl);    {Auxiliary}
        If j=22 then Str(KW[lv]:19:14,Zahl);    {Auxiliary}
        If j=23 then Str(KX[lv]:19:14,Zahl);    {Auxiliary}
        If j=24 then Str(KY[lv]:19:14,Zahl);    {Auxiliary}
        If j=25 then Str(KZ[lv]:19:14,Zahl);    {Auxiliary}
        For k:=1 to Length(Zahl) do
        begin   {use "commata" instead of decimal points}
          If Zahl[k]<>'.' then write(fout,Zahl[k]);
          If Zahl[k]='.' then write(fout,',');
        end;
        Write(fout,chr(9));  {Data-separation Tabulator}
      end;
```

```
      Writeln(fout,'');    {line-separation}
    end;
  end;
  Close(fout);
end;

Function Sgn(Zahl:Integer):Double;
Var merk : Double;
begin
  merk:=0;
  If Zahl>0 then merk:=+1;
  If Zahl<0 then merk:=-1;
  Sgn:=merk;
end;

Procedure Magnetfeld_zuweisen_01; {homogeneous Magnetic field}
Var i,j,k : Integer;
begin
  For i:=-Bn to Bn do  {in x-direction}
  begin
    For j:=-Bn to Bn do  {in y-direction}
    begin
      For k:=-Bn to Bn do  {in z-direction}
      begin
        Bx[i,j,k]:=0.0; {Telsa}
        By[i,j,k]:=1.0; {Telsa}
        Bz[i,j,k]:=0.0; {Telsa}
        OrtBx[i,j,k]:=i*Bsw;
        OrtBy[i,j,k]:=j*Bsw;
        OrtBz[i,j,k]:=k*Bsw;
      end;
    end;
  end;
end;

Procedure Magnetfeld_zuweisen_02; {arbitrary trial of inhomogeneous magnetic field}
Var i,j,k : Integer;
begin
  For i:=-Bn to Bn do  {in x-direction}
  begin
    For j:=-Bn to Bn do  {in y-direction}
    begin
      For k:=-Bn to Bn do  {in z-direction}
      begin
        Bx[i,j,k]:=-Sgn(i)/(i*i+j*j+k*k+1); If i=0 then Bx[i,j,k]:=0; {Telsa}
        By[i,j,k]:=     10/(i*i+j*j+k*k+1);                          {Telsa}
        Bz[i,j,k]:=-Sgn(k)/(i*i+j*j+k*k+1); If k=0 then Bz[i,j,k]:=0; {Telsa}
        OrtBx[i,j,k]:=i*Bsw;
        OrtBy[i,j,k]:=j*Bsw;
        OrtBz[i,j,k]:=k*Bsw;
{       Writeln('Ort:',OrtBx[i,j,k]:12:8,', ',OrtBy[i,j,k]:12:8,', ',OrtBz[i,j,k]:12:8); Wait; }
      end;
    end;
  end;
end;

Procedure Magnetfeld_zuweisen_03;
Var KRPx,KRPy,KRPz : Double; {Cartesian components of the outer product in the counter}
    lmsbetrag      : Double; {absolute value in the denominator}
    lmsbetraghoch3 : Double; {control variable}
    qwill          : Double; {electrical charge, arbitrarily nach S.7}
    om             : Double; {frequency for adjustment qwill to I}
    t              : Double; {time as control variable  0 ... 2*pi/om}
    sx,sy,sz       : Double; {position, where the field has to be determined}
    dHx,dHy,dHz    : Double; {Infinitesimal field element of Biot-Savert}
    Hx,Hy,Hz       : Double; {total field at the point of interest}
    dphi           : Double; {mesh generation of the coil}
    Hxkl,Hykl,Hzkl : Double; {classical result compared}
    Nenner         : Double; {helping variable for classical computation}
    i2,j2,k2       : Integer;{controlled variable for space}
    BXmax,BYmax,BZmax : Double; {field maximum on Y-axis}
Procedure Berechne_dH;
begin
  KRPx:=-om*MEro*cos(om*t)*(MEyo-sy);
  KRPy:=+om*MEro*cos(om*t)*(MEro*cos(om*t)-sx)+om*MEro*sin(om*t)*(MEro*sin(om*t)-sz);
  KRPz:=-om*MEro*sin(om*t)*(MEyo-sy);
  lmsbetrag:=Sqr(MEro*cos(om*t)-sx)+Sqr(MEyo-sy)+Sqr(MEro*sin(om*t)-sz);
  lmsbetrag:=Sqrt(lmsbetrag);
```

```
  lmsbetraghoch3:=lmsbetrag*lmsbetrag*lmsbetrag;
  If lmsbetraghoch3<=1E-50 then begin dHx:=0; dHy:=0; dHz:=0; end;
  If lmsbetraghoch3>=1E-50 then
  begin
    dHx:=qwill*KRPx/4/pi/lmsbetraghoch3*dphi/2/pi;
    dHy:=qwill*KRPy/4/pi/lmsbetraghoch3*dphi/2/pi;
    dHz:=qwill*KRPz/4/pi/lmsbetraghoch3*dphi/2/pi;
  end;
{ Writeln('Infinitesimal Field-element: ',dHx:12:7,', ',dHy:12:7,', ',dHz:12:7,' A/m'); }
end;
Procedure Berechne_Hges;
Var ilok : Integer;{control- variable for coil- meshing}
begin
  Hx:=0;  Hy:=0;  Hz:=0;  {initialisation of the total field}
  qwill:=1;  om:=2*pi*MEI/qwill; {charge and frequency of the magnet emulation coil, *1 von S.7}
  dphi:=2*pi/1000;    {Radiants}
  For ilok:=0 to 999 do  {1000 steps of counting}
  begin
    t:=ilok*dphi/om;  {control  variable (Time), once around the coil}
{   Writeln('ilok = ',ilok:4,' => ',om*t:12:6);  Wait; }
    Berechne_dH;   {Infinitesimal  Field-element of Biot-Savart berechnen}
    Hx:=Hx+dHx;
    Hy:=Hy+dHy;
    Hz:=Hz+dHz;
  end;
{ Writeln('total field at the point of interest. : ',Hx:12:7,', ',Hy:12:7,', ',Hz:12:7,' A/m'); }
  Hxkl:=0; Hzkl:=0; {classic competition for comparison.}
  Nenner:=Sqrt(MEro*MEro+(MEyo-sy)*(MEyo-sy)); Nenner:=2*Nenner*Nenner*Nenner;
  Hykl:=MEI*MEro*MEro/Nenner; {classical comparison is only at y-axis.}
{ Writeln('classical comparison at y-axis: ',Hxkl:12:7,', ',Hykl:12:7,', ',Hzkl:12:7,' A/m'); }
end;
begin
  Writeln; Writeln('Magnetfeld Emulations-Spulenpaar nach *1 von S.5');
  Writeln('y-Koordinaten der Magnetfeld-Emulationsspulen nach *1 von S.5: ',MEyo:8:5,' m');
  Writeln('Radius der Magnetfeld-Emulationsspulen nach *1 von S.5: ',MEro:8:5,' m');
  Writeln('Strom der Magnetfeld-Emulationsspulen nach *1 von S.5: ',MEI:8:5,' Ampere');
  Writeln('Anzahl der Schritte: ',Bn,' hoch 3 => ', 2*Bn+1,' Bildschirm-Aktionspunkte je Spule.');
{ first calculate the top coil: }
  For i2:=-Bn to Bn do  {in x-direction}
  begin
    For j2:=-Bn to Bn do  {in y-direction}
    begin
      For k2:=-Bn to Bn do  {in z-direction}
      begin
        OrtBx[i2,j2,k2]:=i2*Bsw;  sx:=OrtBx[i2,j2,k2];
        OrtBy[i2,j2,k2]:=j2*Bsw;  sy:=OrtBy[i2,j2,k2];
        OrtBz[i2,j2,k2]:=k2*Bsw;  sz:=OrtBz[i2,j2,k2];
        Berechne_Hges;
        Bx[i2,j2,k2]:=muo*Hx; {Telsa}
        By[i2,j2,k2]:=muo*Hy; {Telsa}
        Bz[i2,j2,k2]:=muo*Hz; {Telsa}
{       Write(OrtBx[i2,j2,k2]:10:6,', ',OrtBy[i2,j2,k2]:10:6,', ',OrtBz[i2,j2,k2]:10:6);
        Writeln(' =>',Bx[i2,j2,k2]*1E8:7:4,'E-8, ',By[i2,j2,k2]*1E8:7:4,'E-8, ',Bz[i2,j2,k2]*1E8:7:4,'E-8
                                                                                    Tesla');
        Wait; }
      end;
    end;
    Write('.');
  end;  Writeln(' top coil is calculated.');
{ Writeln('top coil, field at origin of coordinates: ');
  Writeln(Bx[0,0,0],', ',By[0,0,0],', ',Bz[0,0,0]*1E8:7:4,' T'); }
{ Then to add the bottom coil: }
  MEyo:=-MEyo;   {Position of the bottom coil}
  For i2:=-Bn to Bn do  {in x-direction}
  begin
    For j2:=-Bn to Bn do  {in y-direction}
    begin
      For k2:=-Bn to Bn do  {in z-direction}
      begin
        OrtBx[i2,j2,k2]:=i2*Bsw;  sx:=OrtBx[i2,j2,k2];
        OrtBy[i2,j2,k2]:=j2*Bsw;  sy:=OrtBy[i2,j2,k2];
        OrtBz[i2,j2,k2]:=k2*Bsw;  sz:=OrtBz[i2,j2,k2];
        Berechne_Hges;
        Bx[i2,j2,k2]:=Bx[i2,j2,k2]+muo*Hx; {Telsa}
        By[i2,j2,k2]:=By[i2,j2,k2]+muo*Hy; {Telsa}
        Bz[i2,j2,k2]:=Bz[i2,j2,k2]+muo*Hz; {Telsa}
{       Write(OrtBx[i2,j2,k2]:10:6,', ',OrtBy[i2,j2,k2]:10:6,', ',OrtBz[i2,j2,k2]:10:6);
        Writeln(' =>',Bx[i2,j2,k2]*1E8:7:4,'E-8, ',By[i2,j2,k2]*1E8:7:4,'E-8, ',Bz[i2,j2,k2]*1E8:7:4,'E-8
```

```
                                                                                                    Tesla');
        Wait; }
      end;
    end;
    Write('.');
  end;  Writeln(' Untere Spule ist durchgerechnet.'); Writeln;
  MEyo:=-MEyo;   {MEyo reset.}
  Writeln('Gesamtes Feld am Koordinaten-Ursprung: ');
  Writeln(Bx[0,0,0],', ',By[0,0,0],', ',Bz[0,0,0],' T');
  Writeln; Writeln('Gesamtes Feld im Zentrum der oberen Spule:');
  {centre of top coil:} sx:=0; sy:=MEyo; sz:=0;
  Berechne_Hges; BXmax:=muo*Hx; BYmax:=muo*Hy; BZmax:=muo*Hz;
  {centre of bottom coil:} sx:=0; sy:=-MEyo; sz:=0;
  Berechne_Hges; BXmax:=BXmax+muo*Hx; BYmax:=BYmax+muo*Hy; BZmax:=BZmax+muo*Hz;
  Writeln(BXmax,', ',BYmax,', ',BZmax,' T');
  Writeln('Ist dieses Feld gewünscht ? ? ! ? ? ! ? ?');
  Wait; Wait;
end;

Procedure Magnetfeld_anzeigen;
Var i,j,k : Integer;
begin
  Writeln('Feld "Magnetische Induktion" des Dauermagneten:');
  For i:=-Bn to Bn do  {in x-direction}
  begin
    For j:=-Bn to Bn do  {in y-direction}
    begin
      For k:=-Bn to Bn do  {in z-direction}
      begin
        Write('x,y,z=',OrtBx[i,j,k]*100:5:2,', ',OrtBy[i,j,k]*100:5:2,', ',OrtBz[i,j,k]*100:5:2,'cm =>
                                                                                                    B=');
        Write(Bx[i,j,k]:8:4,', ');
        Write(By[i,j,k]:8:4,', ');
        Write(Bz[i,j,k]:8:4,' T ');
        Wait;
      end;
    end;
  end;
end;

Procedure Stromverteilung_zuweisen_03;
Var i : Integer;
begin
  Writeln('Kontrolle der Magnetfeld-Emulations-Spulen:');
  For i:=1 to Round(MESEanz/2) do
  begin
    MESEx[i]:=MEro*cos((i-1)/Round(MESEanz/2)*2*pi); {position of the top field emulation coils}
    MESEy[i]:=MEyo;                                  {position of the top field emulation coils}
    MESEz[i]:=MEro*sin((i-1)/Round(MESEanz/2)*2*pi); {position of the top field emulation coils}
    MESEdx[i]:=-sin((i-1)/Round(MESEanz/2)*2*pi);    {direction of the top field emulation coils}
    MESEdy[i]:=0;                                    {direction of the top field emulation coils}
    MESEdz[i]:=cos((i-1)/Round(MESEanz/2)*2*pi);     {direction of the top field emulation coils}
{   Writeln(i:4,': x,y,z = ',MESEx[i]:12:6 ,', ',MESEy[i]:12:6 ,', ',MESEz[i]:12:6 ,' m');
    Writeln(i:4,': dx,y,z= ',MESEdx[i]:12:6,', ',MESEdy[i]:12:6,', ',MESEdz[i]:12:6,' ');
    Writeln('Laengenkontrolle: ',Sqr(MESEdx[i])+Sqr(MESEdz[i]));  Wait; }
  end;
  For i:=Round(MESEanz/2)+1 to MESEanz do
  begin
    MESEx[i]:=MEro*cos((i-1)/Round(MESEanz/2)*2*pi); {position of the bottom field emulation coils}
    MESEy[i]:=-MEyo;                                 {position of the bottom field emulation coils}
    MESEz[i]:=MEro*sin((i-1)/Round(MESEanz/2)*2*pi); {position of the bottom field emulation coils}
    MESEdx[i]:=-sin((i-1)/Round(MESEanz/2)*2*pi);    {direction of the bottom field emulation coils}
    MESEdy[i]:=0;                                    {direction of the bottom field emulation coils}
    MESEdz[i]:=cos((i-1)/Round(MESEanz/2)*2*pi);     {direction of the bottom field emulation coils}
{   Writeln(i:4,': x,y,z = ',MESEx[i]:12:6 ,', ',MESEy[i]:12:6 ,', ',MESEz[i]:12:6 ,' m');
    Writeln(i:4,': dx,y,z= ',MESEdx[i]:12:6,', ',MESEdy[i]:12:6,', ',MESEdz[i]:12:6,' ');
    Writeln('control of length: ',Sqr(MESEdx[i])+Sqr(MESEdz[i]));  Wait; }
  end;
end;

Procedure Spulen_zuweisen; {coil for the optional input of energy}
Var i,j : Integer;
begin
  {support points of the Polygone:}
  For i:=0 to 2*zo do
  begin   {begin at the left bottom, go first to z- direction}
    SpIx[i+1]:=-xo*Spsw;  SpIy[i+1]:=-yo*Spsw;  SpIz[i+1]:=(i-zo)*Spsw;     {support point}
    SpTx[i+1]:=+xo*Spsw;  SpTy[i+1]:=-yo*Spsw;  SpTz[i+1]:=(i-zo)*Spsw;     {support point}
```

```
    SIx[i+1] :=-xo*Spsw;  SIy[i+1] :=-yo*Spsw;  SIz[i+1] :=(0.5+i-zo)*Spsw; {centre}
    STx[i+1] :=-xo*Spsw;  STy[i+1] :=-yo*Spsw;  STz[i+1] :=(0.5+i-zo)*Spsw; {centre}
    dSIx[i+1]:=0;         dSIy[i+1]:=0;         dSIz[i+1]:=+Spsw;          {direction vector}
    dSTx[i+1]:=0;         dSTy[i+1]:=0;         dSTz[i+1]:=+Spsw;          {direction vector}
  end;
  For i:=0 to 2*yo do
  begin   {then go to y- direction}
    SpIx[2*zo+i+1]:=-xo*Spsw;  SpIy[2*zo+i+1]:=(i-yo)*Spsw;     SpIz[2*zo+i+1]:=+zo*Spsw;  {support point}
    SpTx[2*zo+i+1]:=+xo*Spsw;  SpTy[2*zo+i+1]:=(i-yo)*Spsw;     SpTz[2*zo+i+1]:=+zo*Spsw;  {support point}
    SIx[2*zo+i+1] :=-xo*Spsw;  SIy[2*zo+i+1] :=(0.5+i-yo)*Spsw; SIz[2*zo+i+1] :=+zo*Spsw;  {centre}
    STx[2*zo+i+1] :=+xo*Spsw;  STy[2*zo+i+1] :=(0.5+i-yo)*Spsw; STz[2*zo+i+1] :=+zo*Spsw;  {centre}
    dSIx[2*zo+i+1]:=0;         dSIy[2*zo+i+1]:=Spsw;            dSIz[2*zo+i+1]:=0;    {direction vector}
    dSTx[2*zo+i+1]:=0;         dSTy[2*zo+i+1]:=Spsw;            dSTz[2*zo+i+1]:=0;    {direction vector}
  end;
  For i:=0 to 2*zo do
  begin   {go back and z- direction}
    SpIx[2*zo+2*yo+i+1]:=-xo*Spsw;  SpIy[2*zo+2*yo+i+1]:=yo*Spsw;  SpIz[2*zo+2*yo+i+1]:=(zo-i)*Spsw;
                                                                                    {support point}
    SpTx[2*zo+2*yo+i+1]:=+xo*Spsw;  SpTy[2*zo+2*yo+i+1]:=yo*Spsw;  SpTz[2*zo+2*yo+i+1]:=(zo-i)*Spsw;
                                                                                    {support point}
    SIx[2*zo+2*yo+i+1] :=-xo*Spsw;  SIy[2*zo+2*yo+i+1] :=yo*Spsw;  SIz[2*zo+2*yo+i+1] :=(zo-i-0.5)*Spsw;
                                                                                    {centre}
    STx[2*zo+2*yo+i+1] :=+xo*Spsw;  STy[2*zo+2*yo+i+1] :=yo*Spsw;  STz[2*zo+2*yo+i+1] :=(zo-i-0.5)*Spsw;
                                                                                    {centre}
    dSIx[2*zo+2*yo+i+1]:=0;         dSIy[2*zo+2*yo+i+1]:=0;        dSIz[2*zo+2*yo+i+1]:=-Spsw;
                                                                                    {direction vector}
    dSTx[2*zo+2*yo+i+1]:=0;         dSTy[2*zo+2*yo+i+1]:=0;        dSTz[2*zo+2*yo+i+1]:=-Spsw;
                                                                                    {direction vector}
  end;
  For i:=0 to 2*yo do
  begin   {finally go back in y- direction}
    SpIx[4*zo+2*yo+i+1]:=-xo*Spsw;  SpIy[4*zo+2*yo+i+1]:=(yo-i)*Spsw;      SpIz[4*zo+2*yo+i+1]:=-zo*Spsw;
                                                                                    {support point}
    SpTx[4*zo+2*yo+i+1]:=+xo*Spsw;  SpTy[4*zo+2*yo+i+1]:=(yo-i)*Spsw;      SpTz[4*zo+2*yo+i+1]:=-zo*Spsw;
                                                                                    {support point}
    SIx[4*zo+2*yo+i+1] :=-xo*Spsw;  SIy[4*zo+2*yo+i+1] :=(yo-i-0.5)*Spsw;  SIz[4*zo+2*yo+i+1] :=-zo*Spsw;
                                                                                    {centre}
    STx[4*zo+2*yo+i+1] :=+xo*Spsw;  STy[4*zo+2*yo+i+1] :=(yo-i-0.5)*Spsw;  STz[4*zo+2*yo+i+1] :=-zo*Spsw;
                                                                                    {centre}
    dSIx[4*zo+2*yo+i+1]:=0;         dSIy[4*zo+2*yo+i+1]:=-Spsw;            dSIz[4*zo+2*yo+i+1]:=0;
                                                                                    {direction vector}
    dSTx[4*zo+2*yo+i+1]:=0;         dSTy[4*zo+2*yo+i+1]:=-Spsw;            dSTz[4*zo+2*yo+i+1]:=0;
                                                                                    {direction vector}
  end;   {the very last point is indentically to the first point}
  SpN:=4*zo+4*yo+1;
  Writeln('Anzahl der Punkte der Spulen-Linienaufteilung: von 1 - ',SpN);
  If SpN>SpNmax then
  begin
    Writeln('--- ERROR --- zu viele Spulen-Linienelemente');
    Writeln('--- ABHILFE -> Array groesser dimensionieren');
    Wait; Wait; Halt;
  end;
  {now the area elements:}
  For j:=1 to 2*yo do
  begin
    For i:=1 to 2*zo do
    begin
      FlIx[i+(j-1)*2*zo]:=-xo*Spsw;
      FlIy[i+(j-1)*2*zo]:=(j-0.5-yo)*Spsw;
      FlIz[i+(j-1)*2*zo]:=(i-0.5-zo)*Spsw;
      FlTx[i+(j-1)*2*zo]:=+xo*Spsw;
      FlTy[i+(j-1)*2*zo]:=(j-0.5-yo)*Spsw;
      FlTz[i+(j-1)*2*zo]:=(i-0.5-zo)*Spsw;
    end;
  end;
  FlN:=4*zo*yo;
  Writeln('number of area elements of each coil: von 1 - ',FlN);
  If FlN>FlNmax then
  begin
    Writeln('--- ERROR --- too many area elements');
    Writeln('--- HELP  -> Array should be larger');
    Wait; Wait; Halt;
  end;
end;

Procedure Spulen_anzeigen;  {coil for optional input of energy}
Var i : Integer;
begin
```

```
  Writeln('Input-Sp.-> support point of the Polygon, position, direction vectors:');
  For i:=1 to SpN do
  begin
    Writeln('SP [',i:5,']= ',SpIx[i]*100:10:6,', ',SpIy[i]*100:10:6,', ',SpIz[i]*100:10:6,' cm ');
    Writeln('ORT[',i:5,']= ', SIx[i]*100:10:6,', ', SIy[i]*100:10:6,', ', SIz[i]*100:10:6,' cm ');
    Writeln('RV [',i:5,']= ',dSIx[i]*100:10:6,', ',dSIy[i]*100:10:6,', ',dSIz[i]*100:10:6,' cm ');
    Wait;
  end;
  Writeln('Turbo-Sp.-> support point of the Polygon, position, direction vectors:');
  For i:=1 to SpN do
  begin
    Writeln('SP [',i:5,']= ',SpTx[i]*100:10:6,', ',SpTy[i]*100:10:6,', ',SpTz[i]*100:10:6,' cm ');
    Writeln('ORT[',i:5,']= ', STx[i]*100:10:6,', ', STy[i]*100:10:6,', ', STz[i]*100:10:6,' cm ');
    Writeln('RV [',i:5,']= ',dSTx[i]*100:10:6,', ',dSTy[i]*100:10:6,', ',dSTz[i]*100:10:6,' cm ');
    Wait;
  end;
  Writeln('Input-Spule -> area elements, their centre:');
  For i:=1 to FlN do
  begin
    Write('x,y,z[',i:5,']= ',FlIx[i]*100:10:6,', ',FlIy[i]*100:10:6,', ',FlIz[i]*100:10:6,' cm ');
    Wait;
  end;
  Writeln('Turbo-Spule -> area elements, their centre:');
  For i:=1 to FlN do
  begin
    Write('x,y,z[',i:5,']= ',FlTx[i]*100:10:6,', ',FlTy[i]*100:10:6,', ',FlTz[i]*100:10:6,' cm ');
    Wait;
  end;
  Writeln('-------------------------------------------------------------------------------');
end;

Procedure Magnet_drehen(fi:Double); {rotate by an angle of "fi":}
Var i,j,k : LongInt; {control variables}
begin
  fi:=fi/180*pi;  {go to Radiants}
  For i:=-Bn to Bn do  {x-part}
  begin
    For j:=-Bn to Bn do  {y-part}
    begin
      For k:=-Bn to Bn do  {z-part}
      begin
        {rotation of the position vectors:}
        OrtBxDR[i,j,k]:=+OrtBx[i,j,k]*cos(-fi)+OrtBy[i,j,k]*sin(-fi);
        OrtByDR[i,j,k]:=-OrtBx[i,j,k]*sin(-fi)+OrtBy[i,j,k]*cos(-fi);
        OrtBzDR[i,j,k]:=+OrtBz[i,j,k];
        {rotation of the vectors of field strength:}
        BxDR[i,j,k]:=+Bx[i,j,k]*cos(-fi)+By[i,j,k]*sin(-fi);
        ByDR[i,j,k]:=-Bx[i,j,k]*sin(-fi)+By[i,j,k]*cos(-fi);
        BzDR[i,j,k]:=+Bz[i,j,k];
        {print magnetic field first without rotation and then with rotation:}
{       Write('x,y,z=',OrtBx[i,j,k]:5:2,', ',OrtBy[i,j,k]:5:2,', ',OrtBz[i,j,k]:5:2,'mm => B=');
        Write(Bx[i,j,k]:8:4,', ');
        Write(By[i,j,k]:8:4,', ');
        Write(Bz[i,j,k]:8:4,' T ');  Writeln;
        Write('x,y,z=',OrtBxDR[i,j,k]:5:2,', ',OrtByDR[i,j,k]:5:2,', ',OrtBzDR[i,j,k]:5:2,'mm => B=');
        Write(BxDR[i,j,k]:8:4,', ');
        Write(ByDR[i,j,k]:8:4,', ');
        Write(BzDR[i,j,k]:8:4,' T ');
        Wait; Writeln; }
      end;
    end;
  end;
end;

Procedure Feldstaerke_am_Ort_suchen(xpos,ypos,zpos:Double);
                                                    {this is the position to search the field strength}
Var ixo,iyo,izo : Integer;
    ix,iy,iz    : Integer;
    dist,disto  : Double;
begin
  {first find out which field positions most close to xpos,ypos,zpos .}
  ixo:=0; iyo:=0; izo:=0;
  disto:=Sqrt(Sqr(xpos-OrtBxDR[ixo,iyo,izo])+Sqr(ypos-OrtByDR[ixo,iyo,izo])+Sqr(zpos-
OrtBzDR[ixo,iyo,izo]));
{ Writeln('initial distance to origin of coordinates: ',disto*100:1:15,' cm'); }
  For ix:=-Bn to Bn do  {x-search}
  begin
    For iy:=-Bn to Bn do  {y-search}
```

```
    begin
      For iz:=-Bn to Bn do  {z-search}
      begin
        dist:=Sqrt(Sqr(xpos-OrtBxDR[ix,iy,iz])+Sqr(ypos-OrtByDR[ix,iy,iz])+Sqr(zpos-OrtBzDR[ix,iy,iz]));
        If dist<=disto then
        begin
          ixo:=ix; iyo:=iy; izo:=iz;
          disto:=dist;
{                     Write('Position:   ',OrtBxDR[ix,iy,iz]*100:8:5,',     ',OrtByDR[ix,iy,iz]*100:8:5,',
',OrtBzDR[ix,iy,iz]*100:8:5,' cm'); }
{         Writeln(disto);}                      {Wait;}
        end;
      end;
    end;
  end;
{ Writeln('point number (ixo,iyo,izo): ',ixo,',  ',iyo,',  ',izo); }
  {now I will give the magnetic field at this point:}
{       Writeln('Magnetfeld       dort:       ',BxDR[ixo,iyo,izo]:8:4,',       ',ByDR[ixo,iyo,izo]:8:4,',
',BzDR[ixo,iyo,izo]:8:4,' T '); }
  {now I will calculate the magnetic flux through this coil area element:}
  PsiSFE:=BxDR[ixo,iyo,izo]*Spsw*Spsw;      {nach *1 von S.3}
{ Writeln('magnetic flux through this coil area element: ',PsiSFE,' T*m^2'); }
end;

Procedure Gesamtfluss_durch_Input_Spule;  {according to *2 von S.3}
Var i : Integer;
begin
  PsiGES:=0;
  For i:=1 to FlN do
  begin
    Feldstaerke_am_Ort_suchen(FlIx[i],FlIy[i],FlIz[i]);
    PsiGES:=PsiGES+PsiSFE;
  end;
end;

Procedure Gesamtfluss_durch_Turbo_Spule;  {according to *2 von S.3}
Var i : Integer;
begin
  PsiGES:=0;
  For i:=1 to FlN do
  begin
    Feldstaerke_am_Ort_suchen(FlTx[i],FlTy[i],FlTz[i]);
    PsiGES:=PsiGES+PsiSFE;
  end;
end;

Procedure FourierDatenspeicherung(PSIF : Array of Double); {magnetic flux for Fourier-series}
Var i    : Integer;
    fout : Text;
begin
  Assign(fout,'PSIF.DAT'); Rewrite(fout); {File open}
  Writeln('FOURIER - HIER:');
  For i:=0 to 360 do Writeln(fout,PSIF[i]);
  Close(fout);
end;

Procedure FourierEntwicklung;
Var i    : Integer;
    PSIF : Array [0..360] of Double;
    fin  : Text;
    QSplus,QSmitte,QSminus : Double;
    Qanf,Q1p,Q1m,Q2p,Q2m,Q3p,Q3m : Double; {for B1,2,3 - Iteration}
    Q4p,Q4m,Q5p,Q5m : Double; {for B4,5 - Iteration}
    QSminimum : Double; {for minimum search}
    weiter : Boolean;
Function QuadSum1:Double;
Var merk : Double;
    i    : Integer;
begin
  merk:=0;    {'i' is control variable for the angle,  in Grad}
  For i:=0 to 360 do merk:=merk+Sqr(PSIF[i]-B1*sin(i/360*2*pi));
  QuadSum1:=merk;
end;
Function Fourier(t,Ko1,Ko2,Ko3,Ko4,Ko5:Double):Double;
Var merk : Double;
begin         {'t' is control variable for the angle,  in Grad}
  merk:=Ko1*sin(t/360*2*pi);
  merk:=merk+Ko2*sin(2*t/360*2*pi);
```

```
  merk:=merk+Ko3*sin(3*t/360*2*pi);
  merk:=merk+Ko4*sin(4*t/360*2*pi);
  merk:=merk+Ko5*sin(5*t/360*2*pi);
  Fourier:=merk;
end;
Function QuadSum3(Koeff1,Koeff2,Koeff3:Double):Double;
Var merk : Double;
    i    : Integer;
begin
  merk:=0;        {'i' is control variable for the angle,  in Grad}
  For    i:=0    to    360    do    merk:=merk+Sqr(PSIF[i]-Koeff1*sin(i/360*2*pi)-Koeff2*sin(2*i/360*2*pi)-
Koeff3*sin(3*i/360*2*pi));
  QuadSum3:=merk;
end;
Function QuadSum5(Koeff1,Koeff2,Koeff3,Koeff4,Koeff5:Double):Double;
Var merk : Double;
    i    : Integer;
begin
  merk:=0;
  For i:=0 to 360 do    {'i' is control variable for the angle,  in Grad}
  begin
    If PSIF[i]<>0 then merk:=merk+Sqr(PSIF[i]-Fourier(i,Koeff1,Koeff2,Koeff3,Koeff4,Koeff5));
  end;
  QuadSum5:=merk;
end;
begin
  Assign(fin,'PSIF.DAT'); Reset(fin); {File open}
  Writeln('FOURIER - ENTWICKLUNG:');
  For i:=0 to 360 do Readln(fin,PSIF[i]);
  Close(fin);
  B1:=0;   {average value for the first period as starting condition}
  For i:=0 to 180 do B1:=B1+PSIF[i];
  {estimate the order of magnitude of B1:}
  B1:=B1/90;      {writeln('B1 : ',B1);  Wait;}
  {Put B1 into least square fit:}
  Repeat
    B1:=0.99*B1; QSminus:=QuadSum1;
    B1:=B1/0.99; QSmitte:=QuadSum1;
    B1:=1.01*B1; QSplus:=QuadSum1;  B1:=B1/1.01;
    weiter:=false;
    If QSminus<QSmitte then begin B1:=0.99*B1; weiter:=true; end;
    If QSplus<QSmitte  then begin B1:=1.01*B1; weiter:=true; end;
{   Writeln('QS: ',QSminus,', ',QSmitte,', ',QSplus); }
  Until Not(weiter);
  writeln('B1-vorab : ',B1,',  QS = ',QSmitte);
  {printer values for the purpose of control:}
  AnzP:=360; Abstd:=1;
  For i:=0 to 360 do      {'i' is control variable for the angle,  in Grad}
  begin
    Q[i]:=PSIF[i]; Qp[i]:=B1*sin(i/360*2*pi);
  end;
  {Then put B1 & B2 & B3 into the least square feet:}
  {search initial values for B2 :}
  B2:=0;
  B2:=+B1/10;  QSplus:=QuadSum3(B1,B2,0);
  B2:=-B1/10;  QSminus:=QuadSum3(B1,B2,0);
  If QSplus<QSminus then B2:=+B1/10;
  If QSminus<QSplus then B2:=-B1/10;
  {search initial values for B3 :}
  B3:=0;
  B3:=+B1/10;  QSplus:=QuadSum3(B1,B2,B3);
  B3:=-B1/10;  QSminus:=QuadSum3(B1,B2,B3);
  If QSplus<QSminus then B3:=+B1/10;
  If QSminus<QSplus then B3:=-B1/10;
  Writeln('AnfB1,2,3: ',B1:20,' ,  ',B2:20,' ,  ',B3:20);
  {Put B1, B2, B3 into least square feet:}
  Repeat
    {QuadSums:}
    Qanf:=QuadSum3(B1,B2,B3);
    Q1p:=QuadSum3(B1*1.01,B2,B3);          Q1m:=QuadSum3(B1*0.99,B2,B3);
    Q2p:=QuadSum3(B1,B2*1.01,B3);          Q2m:=QuadSum3(B1,B2*0.99,B3);
    Q3p:=QuadSum3(B1,B2,B3*1.01);          Q3m:=QuadSum3(B1,B2,B3*0.99);
    {find smallest QuadSum:}
    QSminimum:=Qanf;
    If Q1p<QSminimum then QSminimum:=Q1p;  If Q1m<QSminimum then QSminimum:=Q1m;
    If Q2p<QSminimum then QSminimum:=Q2p;  If Q2m<QSminimum then QSminimum:=Q2m;
    If Q3p<QSminimum then QSminimum:=Q3p;  If Q3m<QSminimum then QSminimum:=Q3m;
    {adjust coefficients to smallest QuadSumme :}
```

```
    weiter:=false;
    If Q1p=QSminimum then begin B1:=B1*1.01; weiter:=true; end;
    If Q1m=QSminimum then begin B1:=B1*0.99; weiter:=true; end;
    If Q2p=QSminimum then begin B2:=B2*1.01; weiter:=true; end;
    If Q2m=QSminimum then begin B2:=B2*0.99; weiter:=true; end;
    If Q3p=QSminimum then begin B3:=B3*1.01; weiter:=true; end;
    If Q3m=QSminimum then begin B3:=B3*0.99; weiter:=true; end;
{   Writeln('QS: ',QSminimum); }
  Until Not(weiter);
  Writeln('Nun B1 = ',B1:17,',  B2 = ',B2:17,'  B3 = ',B3:17);
  Writeln('Zugehoerige Quadsum: ',Quadsum3(B1,B2,B3));
  {printer values for the purpose of control:}
  For i:=0 to 360 do
  begin
    Qpp[i]:=B1*sin(i/360*2*pi)+B2*sin(2*i/360*2*pi)+B3*sin(3*i/360*2*pi);
  end;
  {delete very noisy points, with more than 75% distance:}
  For i:=0 to 360 do
  begin
    If    Abs(PSIF[i]-(B1*sin(i/360*2*pi)-B2*sin(2*i/360*2*pi)-B3*sin(3*i/360*2*pi)))>Abs(0.75*B1)    then
PSIF[i]:=0;
  end;
  {Dmake Fourier-series with five coefficients:}
  {search start value for B4 :}
  B4:=0;
  B4:=+B1/40;  QSplus:=QuadSum5(B1,B2,B3,B4,0);
  B4:=-B1/40;  QSminus:=QuadSum5(B1,B2,B3,B4,0);
  If QSplus<QSminus then B4:=+B1/40;
  If QSminus<QSplus then B4:=-B1/40;
  {Ssearch start value for B5 :}
  B5:=0;
  B5:=+B1/40;  QSplus:=QuadSum5(B1,B2,B3,B4,B5);
  B5:=-B1/40;  QSminus:=QuadSum5(B1,B2,B3,B4,B5);
  If QSplus<QSminus then B5:=+B1/10;
  If QSminus<QSplus then B5:=-B1/10;
  Writeln('Und B4,5: ',B4:20,' ,  ',B5:20);
  Writeln('Anf Quadsum: ',QuadSum5(B1,B2,B3,B4,B5));
  {Iteration for B1, B2, B3, B4, B5 least square fit:}
  Repeat
    {QuadSums to be calculated:}
    Qanf:=QuadSum5(B1,B2,B3,B4,B5);
    Q1p:=QuadSum5(B1*1.01,B2,B3,B4,B5);        Q1m:=QuadSum5(B1*0.99,B2,B3,B4,B5);
    Q2p:=QuadSum5(B1,B2*1.01,B3,B4,B5);        Q2m:=QuadSum5(B1,B2*0.99,B3,B4,B5);
    Q3p:=QuadSum5(B1,B2,B3*1.01,B4,B5);        Q3m:=QuadSum5(B1,B2,B3*0.99,B4,B5);
    Q4p:=QuadSum5(B1,B2,B3,B4*1.01,B5);        Q4m:=QuadSum5(B1,B2,B3,B4*0.99,B5);
    Q5p:=QuadSum5(B1,B2,B3,B4,B5*1.01);        Q5m:=QuadSum5(B1,B2,B3,B4,B5*0.99);
    {smallest QuadSumme to be searched:}
    QSminimum:=Qanf;
    If Q1p<QSminimum then QSminimum:=Q1p;  If Q1m<QSminimum then QSminimum:=Q1m;
    If Q2p<QSminimum then QSminimum:=Q2p;  If Q2m<QSminimum then QSminimum:=Q2m;
    If Q3p<QSminimum then QSminimum:=Q3p;  If Q3m<QSminimum then QSminimum:=Q3m;
    If Q4p<QSminimum then QSminimum:=Q4p;  If Q4m<QSminimum then QSminimum:=Q4m;
    If Q5p<QSminimum then QSminimum:=Q5p;  If Q5m<QSminimum then QSminimum:=Q5m;
    {adjust coefficients to smallest QuadSumme :}
    weiter:=false;
    If Q1p=QSminimum then begin B1:=B1*1.01; weiter:=true; end;
    If Q1m=QSminimum then begin B1:=B1*0.99; weiter:=true; end;
    If Q2p=QSminimum then begin B2:=B2*1.01; weiter:=true; end;
    If Q2m=QSminimum then begin B2:=B2*0.99; weiter:=true; end;
    If Q3p=QSminimum then begin B3:=B3*1.01; weiter:=true; end;
    If Q3m=QSminimum then begin B3:=B3*0.99; weiter:=true; end;
    If Q4p=QSminimum then begin B4:=B4*1.01; weiter:=true; end;
    If Q4m=QSminimum then begin B4:=B4*0.99; weiter:=true; end;
    If Q5p=QSminimum then begin B5:=B5*1.01; weiter:=true; end;
    If Q5m=QSminimum then begin B5:=B5*0.99; weiter:=true; end;
{   Writeln('QS: ',QSminimum); }
  Until Not(weiter);
  Writeln('Ergebnis: B1 = ',B1:17,',  B2 = ',B2:17,'  B3 = ',B3:17);
  Writeln('          B4 = ',B4:17,',  B5 = ',B5:17);
  Writeln('Endliche Quadsum: ',Quadsum5(B1,B2,B3,B4,B5));
  {print the values for the purpose of control:}
  For i:=0 to 360 do
  begin
    phipp[i]:=Fourier(i,B1,B2,B3,B4,B5)
  end;
  ExcelAusgabe('fourier.dat',6);
end;
```

```
Function FlussI(alpha:Double):Double;
Var merk : Double;  {alpha in 'radiants'.}
begin
  merk:=B1I*sin(alpha);
  merk:=merk+B2I*sin(2*alpha);
  merk:=merk+B3I*sin(3*alpha);
  merk:=merk+B4I*sin(4*alpha);
  merk:=merk+B5I*sin(5*alpha);
  FlussI:=merk;
end;

Function FlussT(alpha:Double):Double;
Var merk : Double;  {alpha in 'radiants'.}
begin
  merk:=B1T*sin(alpha);
  merk:=merk+B2T*sin(2*alpha);
  merk:=merk+B3T*sin(3*alpha);
  merk:=merk+B4T*sin(4*alpha);
  merk:=merk+B5T*sin(5*alpha);
  FlussT:=merk;
end;

Procedure SinusEntwicklung_fuer_Drehmoment;
Var i,j,jmerk : Integer;
    PSIF : Array [0..360] of Double;
    fin  : Text;
    QSalt,QSneu : Double;
    weiter : Boolean;
    Qanf,QB1plus,QB1minus,Qphaseplus,Qphaseminus : Double; {for numerical Iteration}
    QSminimum : Double; {for the search of the least square fit.}
Function QuadSum2(B1lok,phaselok:Double):Double;
Var merk : Double;
    i    : Integer;
begin
  merk:=0;    {'i' control variable for the angle, in Grad}
  For i:=0 to 360 do merk:=merk+Sqr(PSIF[i]-B1lok*sin((i+phaselok)/360*2*pi));
  QuadSum2:=merk;
end;
begin
  Assign(fin,'PSIF.DAT'); Reset(fin); {File open}
  Writeln('FOURIER-series for quick torque computation:');
  For i:=0 to 360 do Readln(fin,PSIF[i]);
  Close(fin);
  B1:=0;   {search initial value for "B1"}
  For i:=0 to 360 do
  begin
    If PSIF[i]>B1 then B1:=PSIF[i];
  end;
  Writeln('Startwert von B1: ',B1); Wait;
  phase:=0; QSalt:=QuadSum2(B1,phase); jmerk:=Round(phase); {search initial value for "phase"}
  For j:=1 to 360 do
  begin
    phase:=j; QSneu:=QuadSum2(B1,phase);
    If QSneu<QSalt then
    begin
      QSalt:=QSneu;
      jmerk:=j;
{     Writeln(phase,' => ',QSalt); Wait; }
    end;
    phase:=jmerk;
  end;
  Writeln('Startwert von phase: ',phase); Wait;
{Nor the exact Iteration of the Parameters:}
  Repeat
    {QuadSums computation:}
    Qanf:=QuadSum2(B1,phase);
    QB1plus:=QuadSum2(B1*1.0001,phase);
    QB1minus:=QuadSum2(B1*0.9999,phase);
    Qphaseplus:=QuadSum2(B1,phase*1.0001);
    Qphaseminus:=QuadSum2(B1,phase*0.9999);
    {find the smallest QuadSumme:}
    QSminimum:=Qanf;
    If QB1plus<QSminimum     then QSminimum:=QB1plus;
    If QB1minus<QSminimum    then QSminimum:=QB1minus;
    If Qphaseplus<QSminimum  then QSminimum:=Qphaseplus;
    If Qphaseminus<QSminimum then QSminimum:=Qphaseminus;
    {adjust coefficients to the smallest QuadSumme:}
    weiter:=false;
```

```
    If QB1plus=QSminimum     then begin B1:=B1*1.0001; weiter:=true; end;
    If QB1minus=QSminimum    then begin B1:=B1*0.9999; weiter:=true; end;
    If Qphaseplus=QSminimum  then begin phase:=phase*1.0001; weiter:=true; end;
    If Qphaseminus=QSminimum then begin phase:=phase*0.9999; weiter:=true; end;
    Writeln('QS: ',QSminimum);
  Until Not(weiter);
  phase:=phase/360*2*pi; {Phase in Radiants}
  B1dreh:=B1;        {amplitude of torque.}
end;

Function Schnell_Drehmoment(winkel:Double):Double;
begin
  Schnell_Drehmoment:=B1dreh*sin(winkel+phase);
end;

Procedure Magfeld_Turbo_Berechnen(rx,ry,rz,Strom:Double);
Var i : Integer;
    sx,sy,sz    : Double;    {position of the conductor loop elements}
    dsx,dsy,dsz : Double;    {direction vectors of the conductor loop elements}
    AnzLSE      : Integer;   {number of the conductor loop elements}
    smrx,smry,smrz : Double; {Differences for the outer product}
    krpx,krpy,krpz : Double; {outer product in Biot-Savart}
    smrbetrhoch3   : Double; {absolute value for the denominator}
    dHx,dHy,dHz    : Double; {Infinitesimal magnetic field}
    Hgesx,Hgesy,Hgesz:Double;{total magnetic field of the input coil}
begin
{ Spulen_anzeigen; }   {Optional subroutine.}
  AnzLSE:=SpN-1;
  If AnzLSE<>4*yo+4*zo then
  begin
    Writeln('something is wrong:');
    Writeln('problem im mesh-generation of turbo coil');
    Writeln('number of support points of the coil, AnzLSE = ',AnzLSE);
    Writeln('But: 4*yo+4*zo = ',4*yo+4*zo);
    Wait; Wait; Halt;
  end;
  {position and direction vectors of the conductor loop elements, Field according to Biot-Savart:}
  Hgesx:=0;  Hgesy:=0;  Hgesz:=0;
  For i:=1 to AnzLSE do
  begin
    sx:=SpTx[i];  sy:=SpTy[i];  sz:=SpTz[i];     {position of the conductor loop elements}
    dsx:=dSTx[i]; dsy:=dSTy[i]; dsz:=dSTz[i];    {direction vectors of the conductor loop elements}
    smrx:=sx-rx;   smry:=sy-ry;   smrz:=sz-rz;   {Differences for the outer product}
    krpx:=dsy*smrz-dsz*smry;   krpy:=dsz*smrx-dsx*smrz;   krpz:=dsx*smry-dsy*smrx; {outer product}
    smrbetrhoch3:=Sqrt(Sqr(smrx)+Sqr(smry)+Sqr(smrz));
    If smrbetrhoch3<Spsw/1000 then
    begin
      Writeln('Mechanical Kollision -> Magnet touches Turbo-coil. STOP.');
      Writeln('area element at   : ',sx:18,', ',sy:18,', ',sz:18,'m.');
      Writeln('Magnet position at: ',rx:18,', ',ry:18,', ',rz:18,'m.');
      Wait; Wait; Halt;
    end;
    smrbetrhoch3:=smrbetrhoch3*smrbetrhoch3*smrbetrhoch3;
                                                {absolute value for the denominator in Biot-Savart}
    dHx:=Strom*krpx/4/pi/smrbetrhoch3;          {Finite magnetic field of the conductor loop elements}
    dHy:=Strom*krpy/4/pi/smrbetrhoch3;
    dHz:=Strom*krpz/4/pi/smrbetrhoch3;
    Hgesx:=Hgesx+dHx;  Hgesy:=Hgesy+dHy;  Hgesz:=Hgesz+dHz; {Summation of all field elements}
  end;       {next line: algebraic sign according to technical current direction.}
  BTx:=-muo*Hgesx*Nturbo;   BTy:=-muo*Hgesy*Nturbo;   BTz:=-muo*Hgesz*Nturbo;
end;

Procedure Magfeld_Input_Berechnen(rx,ry,rz,Strom:Double);
Var i : Integer;
    sx,sy,sz    : Double;    {position of the conductor loop elements}
    dsx,dsy,dsz : Double;    {direction vectors of the conductor loop elements}
    AnzLSE      : Integer;   {number of the conductor loop elements}
    smrx,smry,smrz : Double; {Differences for the outer product}
    krpx,krpy,krpz : Double; {outer productin Biot-Savart}
    smrbetrhoch3   : Double; {absolute value for the denominator}
    dHx,dHy,dHz    : Double; {Infinitesimal magnetic field}
    Hgesx,Hgesy,Hgesz:Double;{total magnetic field of the input coil}
begin
{ Spulen_anzeigen; } {Optional subroutine.}
  AnzLSE:=SpN-1;
  If AnzLSE<>4*yo+4*zo then
  begin
    Writeln('something is wrong:');
```

```
    Writeln('problem im mesh-generation of input coil');
    Writeln('number of support points of the coil, AnzLSE = ',AnzLSE);
    Writeln('but: 4*yo+4*zo = ',4*yo+4*zo);
    Wait; Wait; Halt;
  end;
  {position and direction vectors of the conductor loop elements, Field according to Biot-Savart:}
  Hgesx:=0;  Hgesy:=0;  Hgesz:=0;
  For i:=1 to AnzLSE do
  begin
    sx:=SpIx[i];  sy:=SpIy[i];  sz:=SpIz[i];     {position of the conductor loop elements}
    dsx:=dSIx[i]; dsy:=dSIy[i]; dsz:=dSIz[i];    {direction vectors of the conductor loop elements}
    smrx:=sx-rx;   smry:=sy-ry;   smrz:=sz-rz;   {Differences for the outer product}
    krpx:=dsy*smrz-dsz*smry;   krpy:=dsz*smrx-dsx*smrz;   krpz:=dsx*smry-dsy*smrx; {outer product}
    smrbetrhoch3:=Sqrt(Sqr(smrx)+Sqr(smry)+Sqr(smrz));
    If smrbetrhoch3<Spsw/1000 then
    begin
      Writeln('Mechanical Kollision -> Magnet touches Turbo-coil. STOP.');
      Writeln('area element at   : ',sx:18,', ',sy:18,', ',sz:18,'m.');
      Writeln('Magnet position at: ',rx:18,', ',ry:18,', ',rz:18,'m.');
      Wait; Wait; Halt;
    end;
    smrbetrhoch3:=smrbetrhoch3*smrbetrhoch3*smrbetrhoch3;
                                                    {absolute value for the denominator in Biot-Savart}
    dHx:=Strom*krpx/4/pi/smrbetrhoch3;           {Finite magnetic field of the conduct loop element}
    dHy:=Strom*krpy/4/pi/smrbetrhoch3;
    dHz:=Strom*krpz/4/pi/smrbetrhoch3;
    Hgesx:=Hgesx+dHx;  Hgesy:=Hgesy+dHy;  Hgesz:=Hgesz+dHz; {Summation of the field elements}
  end;       {next line: algebraic sign according to technical current direction.}
  BIx:=-muo*Hgesx*Ninput;   BIy:=-muo*Hgesy*Ninput;   BIz:=-muo*Hgesz;
end;

Function Drehmoment(alpha:Double):Double;  {Argument : angle of the magnet "alpha"}
Var i : Integer; {control variable}
    Idlx,Idly,Idlz : Double; {Cartesian Components of dl-Vektor according (*1 von S.11)}
    Bxlok,Bylok,Bzlok : Double; {lokal magnetic field}
    FLx,FLy,FLz : Double; {Lorentz-force as outer product}
    dMx,dMy,dMz : Double; {torque of every conductor loop element acting on the magnet.}
    MgesX,MgesY,MgesZ : Double; {SUM: total torque acting on the magnetaus (Emulation-coils).}
    rx,ry,rz : Double; {position of the magnet loop elements after rotation}
begin
  MgesX:=0;  MgesY:=0;  MgesZ:=0;
  For i:=1 to MESEanz do
  begin
    {we now begin with the computation of the Lorentz-force of each element of the magnet-Emulation-coils}
    Idlx:=MEI*MESEdx[i]*4*pi*MEro/MESEanz; {element of the magnet-Emulation-coils}
    Idly:=MEI*MESEdy[i]*4*pi*MEro/MESEanz; {element of the magnet-Emulation-coils}
    Idlz:=MEI*MESEdz[i]*4*pi*MEro/MESEanz; {element of the magnet-Emulation-coils}
    {the next is the magnetic field strength at the position of each conductor loop element}
    Magfeld_Input_Berechnen(MESEx[i],MESEy[i],MESEz[i],qpoI); {adjust current}
    Magfeld_Turbo_Berechnen(MESEx[i],MESEy[i],MESEz[i],qpoT); {adjust current}
    Bxlok:=BIx+BTx;  Bylok:=BIy+BTy;  Bzlok:=BIz+BTz;
                                    {local magnetic field at the position of the conductor loop elements}
    {outer product for computation of Lorentz-force:}
    FLx:=Idly*Bzlok-Idlz*Bylok;
    FLy:=Idlz*Bxlok-Idlx*Bzlok;
    FLz:=Idlx*Bylok-Idly*Bxlok;
    {Check the Lorentz-force:}
{   Writeln('Ort: ',MESEx[i],', ',MESEy[i],', ',MESEz[i]);
    Writeln(' dl: ',MESEdx[i],', ',MESEdy[i],', ',MESEdz[i]);
    Writeln('FLo: ',FLx,', ',FLy,', ',FLz);              }
    {transformation of rotation}
    rx:=+MESEx[i]*cos(-alpha)+MESEy[i]*sin(-alpha);
    ry:=-MESEx[i]*sin(-alpha)+MESEy[i]*cos(-alpha);
    rz:=MESEz[i];
    {from their calculate the torque-element, caused by each Lorenzt-force-Element:}
    dMx:=ry*FLz-rz*FLy;           {torque as outer product M = r x F }
    dMy:=rz*FLx-rx*FLz;
    dMz:=rx*FLy-ry*FLx;
    {check the torque:}
{   Writeln('Dreh:',dMx,', ',dMy,', ',dMz); Wait;         }
    MgesX:=MgesX+dMx; {summation of all torque elements gives the total torque.}
    MgesY:=MgesY+dMy; {in cartesian Components}
    MgesZ:=MgesZ+dMz; {due to the orientation of the magnet, only the z-Component is important.}
  end;                {the magnet rotates around the z-Axis.}
{ Writeln('torque:',MgesX:20,', ',Mgesy:20,', ',Mgesz:20); }
  Drehmoment:=MgesZ;
end;
```

```
Procedure Daten_Speichern;
Var fout  : Text;
    i,j,k : Integer;
begin
  Assign(fout,'schonda'); Rewrite(fout);  {File open}
  {first the Parameters:}
  Writeln(fout,Spsw);
  Writeln(fout,xo);
  Writeln(fout,yo);
  Writeln(fout,zo);
  Writeln(fout,Ninput);
  Writeln(fout,Nturbo);
  Writeln(fout,Bsw);
  Writeln(fout,MEyo);
  Writeln(fout,MEro);
  Writeln(fout,MEI);
  {then the Magnetic field:}   {the number of steps is Bn = "Const."}
  For i:=-Bn to Bn do  {in x-direction}
  begin
    For j:=-Bn to Bn do  {in y-direction}
    begin
      For k:=-Bn to Bn do  {in z-direction}
      begin
        Writeln(fout,OrtBx[i,j,k]);
        Writeln(fout,OrtBy[i,j,k]);
        Writeln(fout,OrtBz[i,j,k]);
        Writeln(fout,Bx[i,j,k]);
        Writeln(fout,By[i,j,k]);
        Writeln(fout,Bz[i,j,k]);
      end;
    end;
  end;
  {the coils and the current distribution can be calculated and does not have to be stored.}
  {the torque-Parameters have to be stored:}
  Writeln(fout,B1T);
  Writeln(fout,B2T);
  Writeln(fout,B3T);
  Writeln(fout,B4T);
  Writeln(fout,B5T);
  Writeln(fout,B1I);
  Writeln(fout,B2I);
  Writeln(fout,B3I);
  Writeln(fout,B4I);
  Writeln(fout,B5I);
  Writeln(fout,B1dreh);
  Writeln(fout,phase);
  Writeln(fout,'All data are atored.');
  Close(fout);
end;

Procedure Alte_Parameter_vergleichen;
Var fin  : Text;
    x : Double;  {Parameters for input}
    n : Integer; {Parameters for input}
    i,j,k : Integer;
begin
  Assign(fin,'schonda'); Reset(fin);  {File open}
  {first the Parameters:}
  Readln(fin,x); If x<>Spsw   then begin schonda:=false; Writeln(' Spsw  geaendert'); end;
  Readln(fin,n); If n<>xo     then begin schonda:=false; Writeln('  xo   geaendert'); end;
  Readln(fin,n); If n<>yo     then begin schonda:=false; Writeln('  yo   geaendert'); end;
  Readln(fin,n); If n<>zo     then begin schonda:=false; Writeln('  zo   geaendert'); end;
  Readln(fin,n); If n<>Ninput then begin schonda:=false; Writeln('Ninput geaendert'); end;
  Readln(fin,n); If n<>Nturbo then begin schonda:=false; Writeln('Nturbo geaendert'); end;
  Readln(fin,x); If x<>Bsw    then begin schonda:=false; Writeln('  Bsw  geaendert'); end;
  Readln(fin,x); If x<>MEyo   then begin schonda:=false; Writeln(' MEyo  geaendert'); end;
  Readln(fin,x); If x<>MEro   then begin schonda:=false; Writeln(' MEro  geaendert'); end;
  Readln(fin,x); If x<>MEI    then begin schonda:=false; Writeln('  MEI  geaendert'); end;
  If schonda then Writeln('Die Parameter sind bereits bekannt.');
  If Not(schonda) then
  begin
    Writeln('Die Parameter sind neu. Es beginnt eine neue Vernetzung.');
    Wait; Wait;
  end;
  {then the magnetic field:}   {the number of steps is Bn = "Const."}
  For i:=-Bn to Bn do  {in x-direction}
  begin
    For j:=-Bn to Bn do  {in y-direction}
```

```
    begin
      For k:=-Bn to Bn do  {in z-direction}
      begin
        Readln(fin,OrtBx[i,j,k]);
        Readln(fin,OrtBy[i,j,k]);
        Readln(fin,OrtBz[i,j,k]);
        Readln(fin,Bx[i,j,k]);
        Readln(fin,By[i,j,k]);
        Readln(fin,Bz[i,j,k]);
      end;
    end;
  end;
  Writeln('Das Magnetfeld ist gelesen.');
  {the coils and the current distribution can be calculated and does not have to be stored.}
  {the torque-Parameters have to be stored:}
  Readln(fin,B1T);
  Readln(fin,B2T);
  Readln(fin,B3T);
  Readln(fin,B4T);
  Readln(fin,B5T);
  Readln(fin,B1I);
  Readln(fin,B2I);
  Readln(fin,B3I);
  Readln(fin,B4I);
  Readln(fin,B5I);
  Writeln('the parameters for the computation of the magnetic flux are read.');
  Readln(fin,B1dreh);
  Readln(fin,phase);
  Writeln('the parameters for the computation of the quick computation of the torque are read.');
  Writeln('Data our prepared for the DFEM-Algorithm.');
  Close(fin);
end;

Function U7:Double;       {Input-voltage for the Input-circuit}
Var UAmpl : Double;       {voltage-Amplitude}
    Pulsdauer : LongInt;  {duration of the pulse in units of time steps "dt"}
    Phasenshift : Double; {Phasenshift between reversal point and voltage-pulse}
    Umerk : Double;       {help variable}
begin
  Umerk:=0;          {Initialisation of help variable}
  UAmpl:=6E-6;       {Volts, voltage-Amplitude}
  Pulsdauer:=20;     {duration of the pulse in units of time steps "dt"}
  Phasenshift:=000;  {Phasenshift between reversal point and voltage-pulse}
{ If i<=Pulsdauer then Umerk:=UAmpl;          {if required: Start-pulse}
  If i>=Pulsdauer then        {triggered Pulses during operation}
  begin
    If (i>=iumk+Phasenshift)and(i<=iumk+Pulsdauer+Phasenshift) then
                                              {the Trigger-Signal is orientated on the top reversal point}
    begin Umerk:=UAmpl; end; {apply voltage}
                                                  {alternatively it could be orientated on the zero-point}
  end;
  U7:=Umerk*0; {Now we do not want to apply a energy-suppl, for the engine is a self-running engine.}
end;

Function Reibung_nachregeln:Double;
Var merk:Double;
begin  {Small Hysterese is necessary:}
  merk:=cr; {if I am not out of hysteresis}
  If (phipo/2/pi*60)>1.000001*phipZiel then merk:=cr*1.000001;
                                                    {if engine is too fast, enhance energy extraction}
  If (phipo/2/pi*60)<0.999999*phipZiel then merk:=cr*0.999999;
                                                     {if engine is too slow, reduce energy extraction}
  If (merk<0.8*crAnfang) then merk:=0.8*crAnfang; {avoid too much oscillation in the control}
  If (merk>1.2*crAnfang) then merk:=1.2*crAnfang; {avoid too much oscillation in the control}
  Reibung_nachregeln:=merk;
end;

Begin {main program}
{ Initialisation - data input: }        {use SI-units}
  Writeln('DFEM-Simulation des EMDR-Motors.');
{ constants of nature:}
  epo:=8.854187817E-12{As/Vm}; {Magnetic field constant}
  muo:=4*pi*1E-7{Vs/Am};       {Elektric field constant}
  LiGe:=Sqrt(1/muo/epo){m/s};  Writeln('speed of light c = ',LiGe, ' m/s');
{ For the solution of the differential equations and plot of the results:}
  AnzP:=100000000; {number of time steps in computation}
  dt:=43E-9;   {seconds} {duration of each single time step}
  Abstd:=1;              {plot control during initialisation}
```

```
  PlotAnfang:=0000;      {first point for the Data-Export to Excel}
  PlotEnde:=100000000;   {last point for the Data-Export to Excel}
  PlotStep:=4000;        {step width for the Data-Export to Excel}
{ Both coils, see. drawing fig.1 :} {automatic mesh generation for the coils}
  Spsw:=0.01; {Meters: step width for the automatic mesh generation}
  xo:=0; yo:=6; zo:=5; {in units of Spsw} {Geometry parameters according to figure 1}
  Spulen_zuweisen;     {coil for input energy,optional}
  Ninput:=100;         {number of windings of the input coil}
  Nturbo:=9;           {number of windings of the turbo coil}
  nebeninput:=10;      {windings side-by-side in Input-coil}
  ueberinput:=10;      {layers of windings on top of each other in Input-coil}
  nebenturbo:=3;       {windings side-by-side in Turbo-coil}
  ueberturbo:=3;       {layers of windings on top of each other in Turbo-coil}
  If nebeninput*ueberinput<>Ninput then
  begin Writeln; Writeln('wrong number of windings: Input-Spule impossible !'); Wait; Wait; Halt; end;
  If nebenturbo*ueberturbo<>Nturbo then
  begin Writeln; Writeln('rong number of windings: Turbo-Spule impossible !'); Wait; Wait; Halt; end;
{ Spulen_anzeigen;     {Optional subroutine to check the positions.}
{ Permanent magnet-Emulation:}   Writeln;  {magnetic field can be measured with Hall probe}
  Bsw:=1E-2; {store magnetic fields in steps of centimetres}
  {not emulate the magnetic field of a one tesla magnet.}
  MEyo:=0.05; {y-coordinates of the emulation coils of the magnet}
  MEro:=0.01; {Radius of the emulation coils of the magnet}
  MEI:=15899.87553474; {Amperes, current in the emulation coils of the magnet}
  schonda:=true; Alte_Parameter_vergleichen;
  If Not(schonda) then Magnetfeld_zuweisen_03;  {calculate and display magnetic field}
  Stromverteilung_zuweisen_03; {calculate current distribution in the emulation coils of the magnet}
{ Other technical values:}
  DD:=0.010;     {Meter}    {diameter of the wire of the coil (input & turbo)}
  rho:=1.35E-8;  {Ohm*m}    {specific electrical resistance of copper}
  rhoMag:=7.8E3; {kg/m^3}   {density of the magnet material, iron, Kohlrausch Bd.3}
  CT:=101.7E-6;  {150E-6;}  {Farad}    {capacitor in the Turbo circuit}
  CI:=100E-6;    {Farad}    {capacitor in the input circuit}
{ Other variables for input:}
  Rlast:=0.030;  {Ohm}   {Ohm's load resistor in the Turbo circuit}
  UmAn:=30000;   {U/min} {mechanical initial conditions of angular velocity, rotating magnet}
  Uc:=0;{Volt}  Il:=0; {Ampere} {electrical initial conditions - no voltage, no current}

{ Mechanical power extraction (the torque is proportional to the angular velocity of the rotating magnet)}
  crAnfang:=45E-6;  {coefficient for mechanical power extraction}
  phipZiel:=30100;  {angular velocity for control of power extraction}

{ Calculated parameters, no input possible:}
  DLI:=4*(yo+zo)*Spsw*Ninput; {Meter} {length of the wire, Input-coil}
  DLT:=4*(yo+zo)*Spsw*Nturbo; {Meter} {length of the wire, Turbo-coil}
  RI:=rho*(DLI)/(pi/4*DD*DD); {Ohm} {Ohm`s resistance of the Input-coil}
  RT:=rho*(DLT)/(pi/4*DD*DD); {Ohm} {Ohm`s resistance of the Turbo-coil}
  BreiteI:=nebeninput*DD; HoeheI:=ueberinput*DD; {width and height of Input-coil}
  BreiteT:=nebenturbo*DD; HoeheT:=ueberturbo*DD; {width and height of Turbo-coil}
  fkI:=Sqrt(HoeheI*HoeheI+4/pi*2*yo*2*zo)/HoeheI; {correction of Induktivity short Input-coil}
  fkT:=Sqrt(HoeheT*HoeheT+4/pi*2*yo*2*zo)/HoeheT; {correction of Induktivity short Turbo-coil}
  Writeln('Induktivitaets-Korrektur: fkI = ',fkI:12:5,',  fkT = ',fkT:12:5);
  LI:=muo*(2*yo+BreiteI)*(2*zo+BreiteI)*Ninput*Ninput/(HoeheI*fkI);
                                                  {Geometrical average => Induktivity Input-coil}
  LT:=muo*(2*yo+BreiteT)*(2*zo+Breitet)*Nturbo*Nturbo/(HoeheT*fkT);
                                                  {Geometrical average => Induktivity Turbo-coil}
  omT:=1/Sqrt(LT*CT);  {circular resonance frequency of the turbo-circuit}
  TT:=2*pi/omT;        {period of the turbo-circuit}
  Mmag:=rhoMag*(pi*MEro*MEro)*(2*MEyo);{Mass of the Magnet}
  J:=Mmag/4*(MEro*MEro+4*MEyo*MEyo/3); {moment of inertia of rotation of the magnet, Dubbel S.B-32}
{ Also to be calculated from the above parameters:}
  omAn:=UmAn/60*2*pi;  {rotating Magnet: angular velocity (rad/sec.), initial value}
  UmSec:=UmAn/60;      {rotating Magnet: rounds per second, initial value}

{ Print the values on the screen:}
  Writeln('*******************************************************************************');
  Writeln('Display few Parametes:');
  Writeln('length of the wire, Input-coil: ',DLI,' m');
  Writeln('length of the wire, Turbo-coil: ',DLT,' m');
  Writeln('Ohm`s resistance of the Input-coil: RI = ',RI:8:2,' Ohm');
  Writeln('Ohm`s resistance of the Turbo-coil: RT = ',RT:8:2,' Ohm');
  Writeln('Induktivity of the Input-coil, ca.: LI = ',LI,' Henry');
  Writeln('Induktivity of the Turbo-coil, ca.: LT = ',LT,' Henry');
  Writeln('circular resonance frequency of the turbo-circuit: omT = ',omT:8:4,' Hz (omega)');
  Writeln('=> period of the turbo-circuit TT = 2*pi/omT = ',TT:15,'sec.');
  Writeln('Magnet: initial angular velocity.: omAn = ',omAn,' rad/sec');
  Writeln('Magnet: initial angular velocity, Umdr./sec.: UmSec = ',UmSec:15:10,' Hz');
  Writeln('Mass of the Magnet = ',Mmag:10:6,' kg');
```

```
  Writeln('moment of inertia of rotation of the magnet',J,' kg*m^2');
  Writeln('total duration of observation: ',AnzP*dt,' sec.');
  Writeln('Excel-Export: ',PlotAnfang*dt:14,'...',PlotEnde*dt:14,' sec., Step ',PlotStep*dt:14,' sec.');
  Writeln('these are ',(PlotEnde-PlotAnfang)/PlotStep:8:0,' Data-lines.');
  If ((PlotEnde-PlotAnfang)/PlotStep)>AnzPmax then
  begin
    Writeln; Writeln('ERROR: too many data-lines.');
    Writeln('so many data-lines cannot be printed in Excel.');
    Writeln('=> stop computation.');  Wait; Wait; Halt;
  end;
{ Wait; }
{ For the preparation, I need AnzP=360, later I will restore the original value.}
  AnzPmerk:=AnzP; {don't forget the value for later}
  AnzP:=360;      {one around in steps of angle-Grad}

{ Test the Data-Export-Routine to Excel:}
  For i:= 1 to AnzP do
  begin
    Q[i]:=i*dt; Qp[i]:=2*i*dt; Qpp[i]:=3*i*dt;      phi[i]:=4*i*dt; phip[i]:=5*i*dt; phipp[i]:=6*i*dt;
    KG[i]:=7*i; KH[i]:=8*i; KI[i]:=9*i; KJ[i]:=10*i; KK[i]:=11*i; KL[i]:=12*i; KM[i]:=13*i; KN[i]:=14*i;
  end;
  {ExcelAusgabe('test.dat',14);}   {Optional subroutine for data export to Excel.}
  {Reset all arrays}
  For i:= 1 to AnzP do
  begin
    Q[i]:=0; Qp[i]:=0; Qpp[i]:=0;      phi[i]:=0; phip[i]:=0; phipp[i]:=0;
    KG[i]:=0; KH[i]:=0; KI[i]:=0; KJ[i]:=0; KK[i]:=0; KL[i]:=0; KM[i]:=0; KN[i]:=0;
  end;

{ Begin on the computation.}
  {Part 1: test of the torque acting on the magnet:}
  Writeln;  {first calculate the magnetic field of the coils (input and turbo)}
  Writeln('first calculate the magnetic field of the coils (input and turbo)');
  Magfeld_Input_Berechnen(-0.00,0.01,0.01,1.0);
                                    {three cartesian components for the position, current = 1.0 Ampere}
  Writeln('B_Input_x,y,z:',BIx:19,', ',BIy:19,', ',BIz:19,' T');
  Magfeld_Turbo_Berechnen(+0.00,0.01,0.01,1.0);
                                    {three cartesian components for the position, current = 1.0 Ampere}
  Writeln('B_Turbo_x,y,z:',BTx:19,', ',BTy:19,', ',BTz:19,' T');
  merk:=Sqrt((2*yo*Spsw*2*zo*Spsw)+Sqr(xo*Spsw)); merk:=merk*merk*merk;
  Writeln('Vgl->Input:    Round    conductor    loop,    Field    in    the    origin    of    coordinates:
',muo*Ninput*1.0*2*yo*Spsw*2*zo*Spsw/2/merk,' T');
  Writeln('Vgl->Turbo:    Round    conductor    loop,    Field    in    the    origin    of    coordinates:
',muo*Nturbo*1.0*2*yo*Spsw*2*zo*Spsw/2/merk,' T');
  {the computation of the both coils (Input & Turbo) is now verified.}
  If Not(schonda) then
  begin         {only for the purpose of control}
    For i:=0 to 360 do
    begin
      KN[i]:=Drehmoment(i/180*pi);
      Writeln(i:4,'Grad => Drehmoment-Komponente: Mz = ',KN[i]);
                                        {The Argument is the angle of the magnet's orientation "alpha"}
    end;
    ExcelAusgabe('drehmom.dat',14);   {Optionales subroutine for data-export to Excel.}
    Writeln('the calculation of the torque is done.');
  end;

  {part 2: test the magnetic flux, which the magnet brings into the coils (to be used later for the
                                                                                    induced voltage)}
  If Not(schonda) then
  begin
    Writeln('we will now calaulate the magnetic flux of geometry "03"');
    Magnet_drehen(00); {angle in Grad , 0...360}
    Gesamtfluss_durch_Input_Spule;     Writeln('total flux in Input-coil: ',PsiGES,' T*m^2');
    Magnet_drehen(01); {angle in Grad , 0...360}
    Gesamtfluss_durch_Input_Spule;     Writeln('total flux in Input-coil: ',PsiGES,' T*m^2');
    Writeln('------------------------');
    Magnet_drehen(00); {angle in Grad , 0...360}
    Gesamtfluss_durch_Turbo_Spule;     Writeln('total flux in Turbo-coil: ',PsiGES,' T*m^2');
    Magnet_drehen(01); {angle in Grad , 0...360}
    Gesamtfluss_durch_Turbo_Spule;     Writeln('total flux in Turbo-coil: ',PsiGES,' T*m^2');
    Writeln('------------------------');
  end;
  {result up to now: the flux difference allows the computation of the induced voltage}

{ Test: rotate the magnet once and measure the magnetic flux and the induced voltage:}
  {use 360 time steps = 360*dt = 36 milliseconds per one turn, corresponding to 1666.666 U/min}
  If Not(schonda) then
```

```
  begin
    Writeln('first the Input-coil:');
    For i:= 0 to 360 do  {first try of the Input-coil}
    begin
      phi[i]:=i; {values in Grad}
      Magnet_drehen(phi[i]);  Gesamtfluss_durch_Input_Spule;  {the result is in "PsiGES"}
      PSIinput[i]:=PsiGES; {this is the magnetic flux in the Input-coil}
      Writeln('phi = ',phi[i]:5:1,' grad  =>  magn. total Fluss = ',PSIinput[i],' T*m^2');
      If i=0 then UindInput[i]:=0;
      If i>0 then UindInput[i]:=-Ninput*(PSIinput[i]-PSIinput[i-1])/dt;
      KG[i]:=0; KH[i]:=PSIinput[i]; KI[i]:=UindInput[i];  {Excel-data output}
    end; Writeln('---------------------------------');
    Writeln('Danach die Turbo-Spule:');
    For i:= 0 to 360 do  {now I will try the Turbo-coil}
    begin
      phi[i]:=i; {values in Grad}
      Magnet_drehen(phi[i]);  Gesamtfluss_durch_Turbo_Spule;  {the result is in "PsiGES"}
      PSIturbo[i]:=PsiGES; {this is the magnetic flux in the Turbo-Spule}
      Writeln('phi = ',phi[i]:5:1,' grad  =>  magn. ges. Fluss = ',PSIturbo[i],' T*m^2');
      If i=0 then Uindturbo[i]:=0;
      If i>0 then Uindturbo[i]:=-Nturbo*(PSIturbo[i]-PSIturbo[i-1])/dt;
      KJ[i]:=0; KK[i]:=PSIturbo[i]; KL[i]:=Uindturbo[i];  {Excel-data output}
      KM[i]:=0; KN[i]:=KN[i]; {two empty columns at the end}
    end;
    {now I smooth the numerical noise:}
    FourierDatenspeicherung(PSIturbo);  FourierEntwicklung;
    B1T:=B1; B2T:=B2; B3T:=B3; B4T:=B4; B5T:=B5;
{**}Writeln('Aktuelle Kontrolle der Fourier-Koeffizienten für den Turbo-Fluß:');
{**}writeln(B1T:13,', ',B2T:13,', ',B3T:13,', ',B4T:13,', ',B5T:13);   Wait;
    FourierDatenspeicherung(PSIinput);  FourierEntwicklung;
    B1I:=B1; B2I:=B2; B3I:=B3; B4I:=B4; B5I:=B5;
    {Controll-Output of the flux curve after smoothing to Excel:}
    For i:=0 to 360 do
    begin       {FlussI and FlussT is the smoothed magnetic flux.}
      KJ[i]:=FlussI(i/360*2*pi);     {the angle of the magnet in "Radiants" to Excel.}
      KM[i]:=FlussT(i/360*2*pi);     {the angle of the magnet in "Radiants" to Excel.}
    end;
  end;
  {The computation of the torque absorbs so much of CPU-time, that it should not be done with smaller
                                                                                                step-width.}
  {Thus I develop a Fourier-serious to accelerate the elapsed computer time:}
  If Not(schonda) then
  begin
    qpoT:=1; qpoI:=0; {quick calibration for turbo-coil, 1A, without Input-coil}
    Writeln('Bring the torque into a sinus-expression in order to save computer time later:');
    For i:=0 to 360 do
    begin       {the total torque, which the magnet gets in the field of both coils(Input&Turbo).}
      KN[i]:=Drehmoment(i*2*pi/360); {The angle of the magnet is given in Radiants.}
      Write('.');  {Writeln(KN[i]);}
    end;
    FourierDatenspeicherung(KN);  SinusEntwicklung_fuer_Drehmoment;
    Writeln('Drehmom-Ampl: ',B1dreh,' und Phase: ',phase);
    {Check whether the quick torque determination gives correct results:}
    For i:=0 to 360 do
    begin
      KG[i]:=Schnell_Drehmoment(i*2*pi/360); {The angle of the magnet is given in Radiants.}
    end;
  end;
  {Store Data, if Parameter-Konfiguration is existing:}
  {If Not(schonda) then} Daten_Speichern;
  {Now the preparation work is done.}

  {I wwill now check whether all necessary data arrived in the file "schonda":}
  For i:=0 to 360 do
  begin       {FlussI and FlussT contains the smoothed magnetic flux through the coils.}
    KJ[i]:=FlussI(i*2*pi/360); {magnetic flux through Input-coil, angle of the magnet in Radiants}
    KM[i]:=FlussT(i*2*pi/360); {magnetic flux through Turbo-coil, angle of the magnet in Radiants}
  end;
  For i:=0 to 360 do
  begin
    KG[i]:=Schnell_Drehmoment(i*2*pi/360); {torque acting on the magnet Magnet, angle in Radiants}
  end;
  ExcelAusgabe('kontroll.dat',14);  {Optionales subroutine for Data-export to Excel.}
  {Now restore the original value for the number timesteps for the solution of the differential
                                                                                          equations:}
  AnzP:=AnzPmerk;
  {Now all data are ready to begin the DFEM-algorithm.}
  Writeln('*************************************************************************');
```

```
  {again one initialisation:  reset all arrays for the subroutine "ExcelLangAusgabe":}
  For i:=0 to AnzPmax do
  begin
   Zeit[i]:=0;  Q[i]:=0;     Qp[i]:=0;     Qpp[i]:=0;  QI[i]:=0;  QpI[i]:=0;  QppI[i]:=0;
   phi[i]:=0;   phip[i]:=0;  phipp[i]:=0;  KJ[i]:=0;   KK[i]:=0;  KL[i]:=0;   KM[i]:=0;
   KN[i]:=0;    KO[i]:=0;    KP[i]:=0;     KQ[i]:=0;   KR[i]:=0;  KS[i]:=0;   KT[i]:=0;
   KU[i]:=0;    KV[i]:=0;    KW[i]:=0;     KX[i]:=0;   KY[i]:=0;
  end;

 {initialisation for the search of the maximum values of current-, angular velocity- and voltage, to be
                                                                                    displayed on the screen:}
  QTmax:=0; QImax:=0; QpTmax:=0; QpImax:=0; QppTmax:=0; QppImax:=0; phipomax:=0;
  Wentnommen:=0; {initialisation of the extracted energy at the load resistor}
  Ereib:=0;      {initialisation of the mechanical extracted energy}

  {initialisation of the Reference for the Input-voltage-Signal:}
  steigtM:=false;   steigtO:=false;
  Ezuf:=0;  {initialisation supplied energy by input voltage}
  LPP:=0;   {initialisation number of data points for Excel-Plot}

{ This is now the moment to start the solution of the system of differential equations:}
{ The main core of the computation begins:}
{ We start out with the initial conditions:}
  phio:=0;      phipo:=omAn;  {phippo:=0;} {initial condition of the mechanical rotation of the magnet}
                                           {we start with a given angular velocity}
  qoT:=CT*Uc;   qpoT:=Il;     {qppoT:=0;}  {electrical initial conditions of the Turbo-circuit, here ZERO}
                                           {the capacitor in the Turbo circuit can be pre-charged if
required}
  qoI:=0;       qpoI:=0;       qppoI:=0;   {electrical initial conditions of the Input-circuit, here ZERO}
{ For the step number zero, there is no "step before":}
  {phim:=phio;}{phipm:=phipm;} {phippm:=phippm;}
  qmT:=qoT;    {qpmT:=qpmT;}   {qppmT:=qppmT;}
  {qmI:=qoI;}  {qpmI:=qpmI;}   {qppmI:=qppmI;}

{ Initial conditions are ready, solver begins:}
  For i:=0 to AnzP do
  begin
    {initialisation of the Reference of the Input-voltage-Signal:}
    If i=0 then iumk:=0;
    If i>=1 then {Input-voltage-Reference to be orientated on the turbo circuit.}
    begin
      steigtM:=steigtO;  {remind the signal slope}
      If qoT>qmT then steigtO:=true;
      If qoT<qmT then steigtO:=false;
      If (steigtM)and(Not(steigtO)) then iumk:=i;
    end;
    {Aktual Moment of Analysis, running time in seconds, the moment "now", "Jetzt-Schritt":}
    Tjetzt:=i*dt;
    {the last step will be the step before the moment now:}
    phim:=phio; phipm:=phipo; {phippm:=phippo;}   {rotation}
    qmT:=qoT; qpmT:=qpoT;     {qppmT:=qppoT;}     {Turbo-coil}
    qmI:=qoI; qpmI:=qpoI; qppmI:=qppoI;           {Input-coil}
    {the new step will be calculated as following:}
    {first the rotation of the magnets, talk is generated by the current in the coils:}
{KK}phippo:=Schnell_Drehmoment(phim)*qpoT/J;
        {subroutine "Schnell_Drehmoment" is scaled with ITurbo=1A & IInput=0A, linear with Turbo-current.}
{!! all line with "!!" are commented out because the input coil is not used.}
{!! phippo:=Drehmoment(phim)/J;
                              {Complete torque-computation with Turbo-coil and Input-coil, not very fast.}
{For phippo -> I have two alternatives depending on whether the input-coil is active or not.}
{if the input coil is active, I shall allways set "schonda:=false", so that the complete preparation is
                                                                                    computed for every run.}
{All "GG"-line are for extraction of mechanical power:}
{GG}If i=1 then cr:=crAnfang;            {coefficient of friction proportional to angular velocity}
{GG}If i>1 then cr:=Reibung_nachregeln;
             {the coefficient of friction can be controlled in order to keep the angular velocity stable.}
{GG}If phipo>0 then phippo:=phippo-cr*phipm/J; {negative acceleration acts against the angular velocity}
{GG}If phipo=0 then phippo:=phippo;
{GG}If phipo<0 then phippo:=phippo+cr*phipm/J; {negative acceleration acts against the angular velocity}
{GG}{now friction respectively energy extraction is calculated.}
    If (i mod 100000)=0 then write('.');
    phipo:=phipm+phippo*dt;             {1. step of integration without extracting mechanical power}
    phio:=phim+phipo*dt;                {2. step of integration}
{GG}Preib:=cr*phipm*phipo;       {extracting mechanical power now}
{GG}Ereib:=Ereib+Preib*dt;       {computation of the power and energy being extracted}
    {Dann die Turbo-Spule. Gedämpfte elektrische Schwingung, dazu induzierte Spannung aufgrund Magnet-
                                                                                    Drehung:}
```

```
{FF}qppoT:=-1/(LT*CT)*qmT-(RT+Rlast)/LT*qpmT; {differential equation of the attenuated oscillation.}
    UinduzT:=-Nturbo*(FlussT(phio)-FlussT(phim))/dt;
                                          {bring the induced voltage into the differential equations}
    qppoT:=qppoT-UinduzT/LT;                  {the induced voltage acts on the second time derivative}
{??}qpoT:=qpmT+qppoT*dt; {-Rlast/(2*LT)*qpmT*dt;} {1. step of integration}
    qoT:=qmT+qpoT*dt;                          {2. step of integration}
    {Dann die Input-Spule:}     UinduzI:=0;
    qoI:=qmI; qpoI:=qpmI; qppoI:=qppmI;          {The input coil doesn't do anything now.}
{!! If I want to activate the input coil, I shall activate the following five lines:}
{!! qppoI:=-1/(LI*CI)*qmI-RI/LI*qpmI+U7/LI;
                {differential equation of attenuated oscillation, perturbation function for Input-voltage}
{!! UinduzI:=-Ninput*(FlussI(phio)-FlussI(phim))/dt;
                                          {the induced voltage acts of the rotation of the magnet}
{!! qppoI:=qppoI-UinduzI/LI;
                          {action of the induced voltage on the second derivative of q, namely "qppoT"}
{!! qpoI:=qpmI+qppoI*dt;                    {1. step of integration}
{!! qoI:=qmI+qpoI*dt;                       {2. step of integration}
    Pzuf:=U7;{*qpoI}                        {supplied power by input voltage}
    Ezuf:=Ezuf+Pzuf*dt;                     {supplied energy babe would voltage}
{caution: the quick torque-computation "phippo" does not work for Turbo- & Input-coil. The current is not
                                                                                known in the subroutines.}

    {I now one to find the maximum values for current, voltage and angular velocity:}
    If Abs(qoT)>QTmax then QTmax:=Abs(qoT);  {Maximum of electrical charge in the Turbo capacitor}
    If Abs(qoI)>QImax then QImax:=Abs(qoI);  {Maximum of electrical charge in the Input capacitor}
    If Abs(qpoT)>QpTmax then QpTmax:=Abs(qpoT);       {Maximum of electrical current in the Turbo coil}
    If Abs(qpoI)>QpImax then QpImax:=Abs(qpoI);       {Maximum of electrical current in the Input coil}
    If Abs(qppoT)>QppTmax then QppTmax:=Abs(qppoT);   {Maximum of Ipunkt in Turbo-coil}
    If Abs(qppoI)>QppImax then QppImax:=Abs(qppoI);   {Maximum of Ipunkt in  Input-coil}
    If Abs(phipo)>phipomax then phipomax:=Abs(phipo); {Maximum of angular velocity of the magnet}
    Wentnommen:=Wentnommen+Rlast*qpoT*qpoT*dt;  {summation of the extracted energy at the load resistor}

    {now export the data into Excel:}
    If (i>=PlotAnfang)and(i<=PlotEnde) then {These lines shall be plotted to Excel.}
    begin
      If ((i-PlotAnfang)mod(PlotStep))=0 then
      begin
        znr:=Round((i-PlotAnfang)/PlotStep);
        Zeit[znr]:=Tjetzt;                                    {time-scale.}
        Q[znr]:=qoT;    Qp[znr]:=qpoT;    Qpp[znr]:=qppoT;
                                        {Turbo-coil, it Array without index "T" (and only there!).}
        QI[znr]:=qoI;   QpI[znr]:=qpoI;   QppI[znr]:=qppoI;   {Input-coil}
        phi[znr]:=phio; phip[znr]:=phipo; phipp[znr]:=phippo; {rotation of the magnet}
        KK[znr]:=FlussT(phio);    KL[znr]:=FlussI(phio);      {magnetic flux through the coils}
        KM[znr]:=UinduzT;         KN[znr]:=UinduzI;           {voltage induced into the coils}
        KO[znr]:=1/2*LT*qpoT*qpoT;                            {Energy in Input-coil}
        KP[znr]:=1/2*LI*qpoI*qpoI;                            {Energy in Turbo-coil}
        KQ[znr]:=1/2*qoT*qoT/CT;                              {Energy im Input-capacitor}
        KR[znr]:=1/2*qoI*qoI/CI;                              {Energy im Turbo-capacitor}
        KS[znr]:=1/2*J*phipo*phipo;                           {Energy of Magnet-Rotation}
        KT[znr]:=KO[znr]+KP[znr]+KQ[znr]+KR[znr]+KS[znr];     {total energy in the system}
        KU[znr]:=Rlast*qpoT*qpoT;      {power being extracted at the load resistor in the Turbo circuit}
        KV[znr]:=U7;                   {control of the input voltage in the input circuit}
        KW[znr]:=Pzuf;                 {power supply by the input voltage}
        KX[znr]:=cr;         {control of the coefficient of friction for mechanical extraction of power}
        KY[znr]:=Preib;
                 {mechanical power being extracted, emulated by friction proportional to angular velocity}
        KZ[znr]:=0;       {one column for optional data, not used now.}
        LPP:=znr;         {number of the last plot point, length of the excel-file -> ExcelLangAusgabe}
      end;
    end;
    AnfEnergie:=KU[0];         {initial total energy within the system}
    EndEnergie:=KU[LPP];       {total energy at the end of observation within the system}
  end;
  Writeln; Writeln('number of data points for Excel-Plot: LPP = ',LPP);
  Writeln; Writeln('display of some amplitudes Amplitudes: (not effective-values)');
  Writeln('Input-capacitor, voltage, UmaxI =',QImax/CI,' Volt');
                                                    {Maximum of electrical charge in Input-capacitor}
  Writeln('Turbo-capacitor, voltage, UmaxT =',QTmax/CT,' Volt');
                                                    {Maximum of electrical charge in Turbo-capacitor}
  Writeln('Input-circuit,   current, ImaxI =',QpImax,' Ampere');
                                                      {Maximum of electrical current in Input-coil}
  Writeln('Turbo-circuit,   current, ImaxT =',QpTmax,' Ampere');
                                                      {Maximum of electrical current in Turbo-coil}
  Writeln('Input-coil,      voltage, UmaxSI=',LI*QppImax,' Volt');
                                                                {Maximum of Ipunkt in der Input-coil}
  Writeln('Turbo-coil,      voltage, UmaxST=',LT*QppTmax,' Volt');
                                                                {Maximum of Ipunkt in der Turbo-coil}
```

```
  Writeln('Maximum of angular velocity  =   ',phipomax,' rad/sec');
                                                {Maximum of the angular velocity of the magnet}
  Writeln('Maximum of angular velocity  =   ',phipomax/2/pi*60:15:6,' U/min.');
                                                {Maximum of the angular velocity of the magnet}
  Writeln('angular velocity at the end  =   ',phip[LPP]/2/pi*60:15:6,' U/min.');
  Writeln;
  Writeln('initial energy in the System:    ',AnfEnergie:18:11,' Joule');
  Writeln('system`s energy at the end:      ',EndEnergie:18:11,' Joule');
  Writeln('Energy-gain during observation:  ',(EndEnergie-AnfEnergie):18:11,' Joule');
  Writeln('Power-gain during observation :  ',(EndEnergie-AnfEnergie)/(AnzP*dt):18:11,' Watt');
  Writeln('total energy extracted at load resistor = ',Wentnommen:18:11,' Joule');
  Writeln('corresponding to a power of :    ',Wentnommen/(AnzP*dt):18:11,' Watt');
  Writeln('supplied energy by input supply: ',Ezuf,' Joule');
  Writeln('corresponding to a power of:     ',Ezuf/(AnzP*dt),' Watt');
  Writeln('total extracted mechanic. energy:',Ereib:18:11,' Joule');
  Writeln('corresponding to a power of  =   ',Ereib/(AnzP*dt):18:11,' Watt');
  Writeln('total duration of observation :  ',(AnzP*dt):18:11,' sec.');
  ExcelLangAusgabe('test.dat',25);
  Writeln; Writeln('computation done -> bye bye.');
  Wait;   Wait;
End.
```