Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 10/04

Die Mathematische Zauberkiste

Mathematik für alle
Mathematische Knobeleien
Zeige mal, was du kannst!

Neundorf, Werner

März, 2010

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69 3621
Fax: +49 3677 69 3270
http://www.tu-ilmenau.de/ifm/

ISSN xxxx-xxxx

ilmedia

Technische Universität Ilmenau
Fakultät für Mathematik
und Naturwissenschaften
Institut für Mathematik
http://www.tu-ilmenau.de/math/

Postfach 10 05 65
D - 98684 Ilmenau
Germany
Tel.: 03677/69 3267
Fax: 03677/69 3272
Telex: 33 84 23 tuil d.
email: werner.neundorf@tu-ilmenau.de

Preprint No. M 04/10

Die Mathematische Zauberkiste

Mathematik für alle
Mathematische Knobeleien
Zeige mal, was du kannst!

Werner Neundorf

März 2010

[‡]MSC (2000): 03-04, 65-01, 65C20, 68W30, 97A80, 97M10, 97Q60

Zusammenfassung

Beobachtungen und Erfahrungen zeigen, dass es für Schüler, Studenten und Interessierte nicht unbedingt notwendig ist, das große Szenarium eines noch größeren Programms nahezubringen. Vielmehr erweisen sich überschaubare Probleme und Algorithmen als wirksame und moderne didaktische Werkzeuge. Sie gestatten insbesondere die weitgehende Konfigurierbarkeit und damit kognitive Manipulationsmoglichkeiten, auch bezüglich grafischer und algebraisch-symbolischer Elemente, Darstellungsweisen und Umfang der Ergebnisse, Funktions- und Methodenauswahl, interaktive Arbeitsweise und Wiederholbarkeit unter Verwendung von Software und Computern.
Ausgewählte Demonstrationen aus unterschiedlichen Gebieten sind in der vorliegenden **Mathematischen Zauberkiste** beschrieben und zusammengefasst. Nach nützlichen Vorbetrachtungen erfolgen die Berechnungen sowie die zahlreichen grafischen Darstellungen der Funktionen, der Algorithmen und Situationen sowie der Animationen unter Verwendung der Computeralgebrasysteme Maple und MATLAB.
Sie sollen beim Leser das Interesse und die Neugier auf mehr wecken.

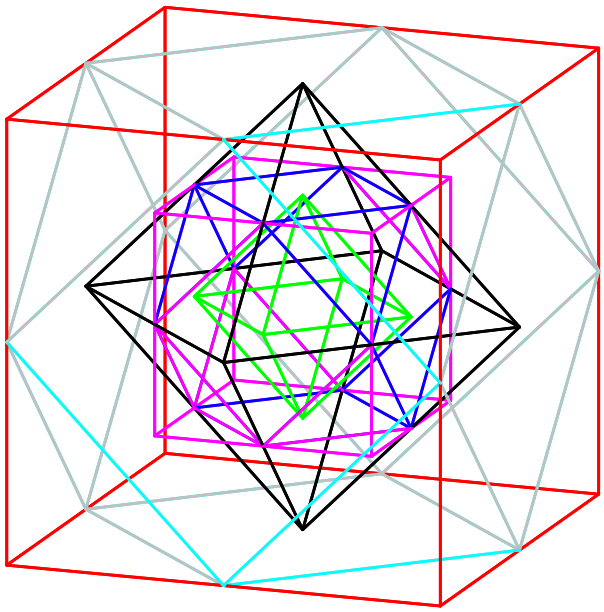
Observations and experiences have shown, that it is not necessary to confront scoolchildren, students or interested peoples with big shells of very extensive software tools. Instead, limited programs prove as effective and modern didactic means. They allow especially the far-reaching changes of configuration and in this way some possibilities of doing cognitive manipulations with regard to graphical and algebraic-symbolic elements, presentation and scope of results, selection of functions and methods, interactive mode of operation and repetition, using software and computers.
This is supported by selected examples and demonstrations from different areas in the presented **Mathematical Magic Box**. After some basic considerations we make the calculations, the graphics of functions, algorithms and situations, some animations by computer algebra software like Maple and MATLAB.
All this has to wake up the interest and the curiosity of the readers.

Godfrey Harold Hardy 1940

Das entscheidende Kriterium ist Schönheit; für hässliche Mathematik ist auf dieser Welt kein beständiger Platz.

Albrecht Beutelspacher

Mathematik ist eine basisdemokratische Wissenschaft.
Jeder kann eine logische Argumentation nachvollziehen.



Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Zur Begrüßung

Maple-Arbeitsblatt zur Begrüßung der Leser

Zugleich ist es das erste Beispiel für das Zusammenspiel von Problem, Idee, Methodenwahl, mathematische Modellierung/Algorithmus, Softwarelösung/Implementation, Simulation und Animation.

```
TU Ilmenau  IfMath  PD Dr. W.Neundorf
Datei:  zauberkiste1.mws

"Die Mathematische Zauberkiste"
----------------------------------------------------------------------
```

```
> restart:
  with(linalg):
  with(plots):
  with(plottools):
```

```
Titel mit Animation einer Lichtschlange
Definition von Schlauch/Torus als parametrische Kurve
```

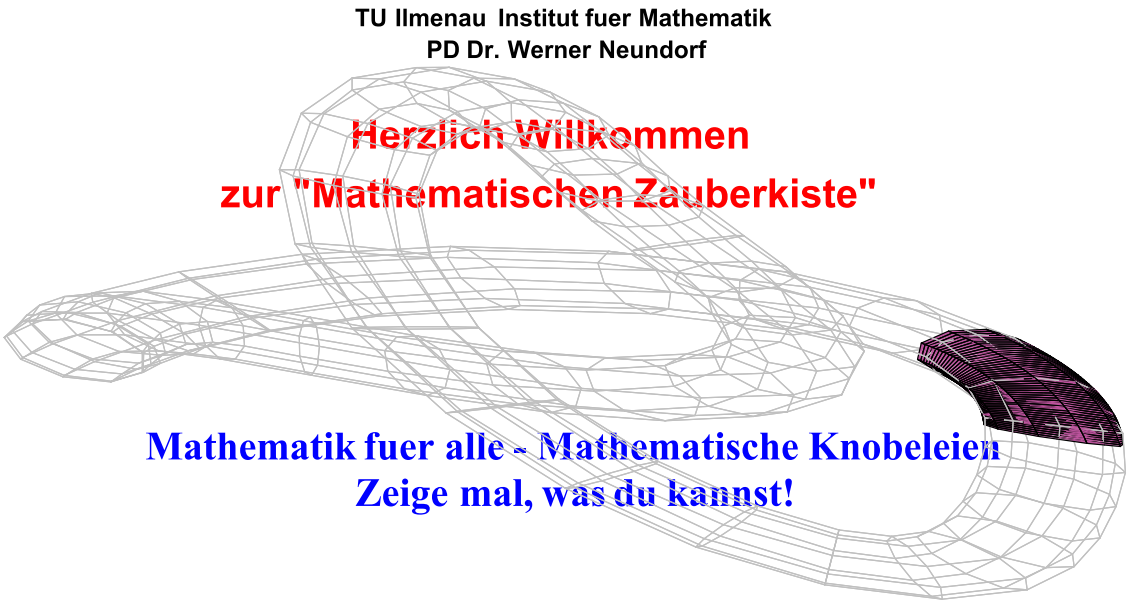
```
> setoptions(scaling=constrained,axes=none):
> curve1:=[-10*cos(t)-2*cos(5*t)+15*sin(2*t),
           -15*cos(2*t)+10*sin(t)-2*sin(5*t),
            10*cos(3*t)]:
```

```
Gesamtgitter anzeigen sowie
Schlauch st"uckweise anzeigen und aneinanderf"ugen
```

```
> d131:=plots[tubeplot](curve1,t=0..2*Pi,radius=3,
                        style=wireframe,color=gray):
  q1:=display(d131):

> d132:=i->plots[tubeplot](curve1,t=2*Pi*(i-1)/n..2*Pi*i/n,radius=3):
  n:=20:
```

Animation der Lichtschlange mit Torusgitter

```
> tp:=textplot3d([[-40,-25,-30,‘Herzlich Willkommen‘],
                  [-40,-25,-40,‘zur "Mathematischen Zauberkiste"‘]],
                  color=red,font=[HELVETICA,BOLD,25]):
  tp1:=textplot3d([[-42,-25,-85,‘Mathematik fuer alle -
                                  Mathematische Knobeleien‘],
                  [-42,-25,-93,‘Zeige mal, was du kannst!‘]],
                  color=blue,font=[TIMES,BOLD,25]):

> for i from 1 to n do
   q2:=display(d132(i),style=patch,
               color=COLOR(RGB,rand()/10^12,rand()/10^12,rand()/10^12)):
   q2z[i]:=plots[display](tp,tp1,q1,q2):
  end do:
  dk13b:=[seq(q2z[i],i=7..n),seq(q2z[i],i=1..6)]:
> display(dk13b,insequence=true,style=patch,
    title=‘TU Ilmenau  Institut fuer Mathematik\nPD Dr. Werner Neundorf‘,
    titlefont=[HELVETICA,BOLD,15],view=[-250..250,-150..150,-280..280]);
```



Hinweise zur Dateiarbeit in Maple beim Speichern von Bildern als File findet man in [7]. So gibt es die Möglichkeit, eine auf dem Bildschirm erzeugte Grafik durch eine implementierte Menüfunktion im gewünschten Format zu exportieren. Weiterhin steht die `interface`-Funktion mit Ausgabeformaten wie `ps, gif, jpeg, pcx` zur Verfügung. Für die obige Figur als animiertes transparentes `gif`-File notiert man

```
datei:=‘D:/neundorf/titel2.gif‘:
interface(plotdevice=gif,plotoutput=datei,
          plotoptions=‘width=400,height=400,transparent=true‘):
display(dk13b,insequence=true,style=patch,
  title=‘TU Ilmenau  Institut fuer Mathematik\nPD Dr. Werner Neundorf‘);
interface(plotdevice=default):
```

1.2 Popularisierung der Mathematik und Naturwissenschaften

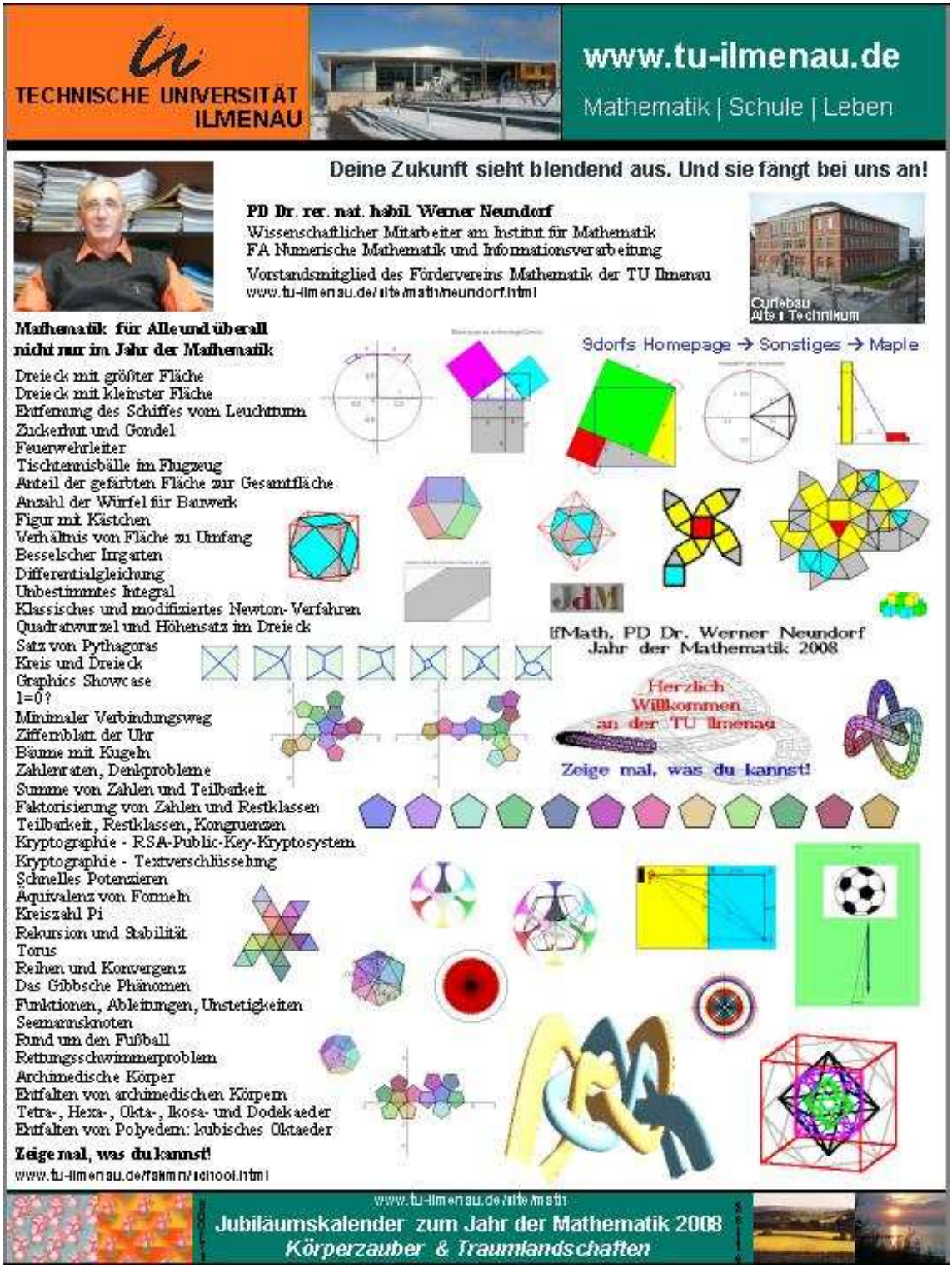
Die Bedeutung der Mathematik in Theorie und Praxis steht außer Zweifel. Auch wenn nicht jeder diese unbedingt wahrnimmt und anerkennt. Deshalb ist jedes Engagement für die Popularisierung der Mathematik wichtig. Dabei sind natürlich insbesondere die Mathematiker gefragt und gefordert. Im Zusammenspiel mit anderen Wissenschaften bieten sich genügend Probleme und vielfältige Aspekte an, mathematisch beleuchtet und untersucht zu werden. Und wenn es dabei gelingt, mit den Fragestellungen auch ein deutliches gesellschaftliches Interesse zu entwickeln, werden die Wechselbeziehungen sowohl den gesellschaftlichen Fortschritt allgemein als auch den disziplinären Erkenntnisstand speziell beförden.
Das Jahr der Mathematik 2008 und der Monat Mai der Mathematik 2009 waren besondere Anlässe, mit verschiedenen Veranstaltungen die Aufmerksamkeit möglichst breiter Bevölkerungsschichten auf Nutzen und Schönheit der Mathematik, auf mathematisches Arbeiten und Problemlösen, auf die großen Erfolge und die zahlreichen Rätsel der Mathematik, aber auch auf die Verantwortung für den sachgerechten Umgang mit mathematischen Aussagen und Methoden zu lenken.
Es war ein schöner Zufall, dass das Institut für Mathematik der TU Ilmenau im Jahr 2008 auch den 40. Jahrestag der Einführung des Studienganges Mathematik beging. Damit gab es für die Angehörigen des Institutes einen weiteren Grund, in Vorträgen, Begegnungen und Diskussionen mit mathematisch interessierten Schülern aus Schulen unserer Region den Reichtum, die Bedeutung und Faszination der Mathematik zu unterstreichen.

Ich habe mich von den Vorträgen inspirieren lassen und mich dabei selbst mit vielfältigen Aktivitäten eingebracht. Ich griff kleinere und größere interessante Probleme aus dem akademischen Alltag und den mathematischen Lehrveranstaltungen auf. Diese betrachtete, untersuchte und bearbeitete ich unter verschiedenen Gesichtspunkten, um letztendlich daraus Softwarelösungen, Demonstrationen, Animationen und attraktive Vortragsangebote zu machen, die Neugier und Mitmachen bei den Zuhörern weckten und wecken.

So entstand unter dem Titel “Zeige mal, was du kannst!“ ein Liste von Aufgabenstellungen mit Lösungen als Maple-Arbeitsblätter oder Programme in anderen Sprachen. Das folgende Poster zum Jahr der Mathematik 2008 gibt einen Einblick in die Vielfalt der behandelten Themen.

Parallel dazu sammelte und konstruierte ich Modelle, Demonstrationsmaterial und -gegenstände. Alle diese Sachen befinden sich in der “Mathematischen Zauberkiste“.

Inzwischen sind sowohl die Aufgabensammlung mit praktischem Anschauungsmaterial und Softwarelösungen als auch die Zauberkiste weiter gewachsen. Ich möchte dem Leser eine kleine Auswahl vorstellen. Die Präsentation aller Themen würde den Rahmen dieser Arbeit sprengen.


TECHNISCHE UNIVERSITÄT ILMENAU
www.tu-ilmenau.de
Mathematik | Schule | Leben
Deine Zukunft sieht blendend aus. Und sie fängt bei uns an!
PD Dr. rer. nat. habil. Werner Neundorf
Wissenschaftlicher Mitarbeiter am Institut für Mathematik
FA Numerische Mathematik und Informationsverarbeitung
Vorstandsmitglied des Fördervereins Mathematik der TU Ilmenau
www.tu-ilmenau.de/site/math/neundorf.html
Curiebau
Altes Technikum
Mathematik für Alle und überall
nicht nur im Jahr der Mathematik
Dreieck mit größter Fläche
Dreieck mit kleinster Fläche
Entfernung des Schiffes vom Leuchtturm
Zuckerhut und Gondel
Feuerwehrleiter
Tischtennisbälle im Flugzeug
Anteil der gefärbten Fläche zur Gesamtfläche
Anzahl der Würfel für Bauwerk
Figur mit Kästchen
Verhältnis von Fläche zu Umfang
Besselscher Irrgarten
Differentialgleichung
Unbestimmtes Integral
Klassisches und modifiziertes Newton-Verfahren
Quadratwurzel und Höhensatz im Dreieck
Satz von Pythagoras
Kreis und Dreieck
Graphics Showcase
1=0?
Minimaler Verbindungsweg
Ziffernblatt der Uhr
Bäume mit Kugeln
Zahlenraten, Denkprobleme
Summe von Zahlen und Teilbarkeit
Faktorisierung von Zahlen und Restklassen
Teilbarkeit, Restklassen, Kongruenzen
Kryptographie - RSA-Public-Key-Kryptosystem
Kryptographie - Textverschlüsselung
Schnelles Potenzieren
Äquivalenz von Formeln
Kreiszahl Pi
Rekursion und Stabilität
Torus
Reihen und Konvergenz
Das Gibbsche Phänomen
Funktionen, Ableitungen, Unstetigkeiten
Seemannsknoten
Rund um den Fußball
Rettungsschwimmerproblem
Archimedische Körper
Entfalten von archimedischen Körpern
Tetra-, Hexa-, Okta-, Ikosa- und Dodekaeder
Entfalten von Polyedern: kubisches Oktaeder
Zeige mal, was du kannst!
www.tu-ilmenau.de/fakmn/school.html
9dorfs Homepage → Sonstiges → Maple
JdM
IfMath, PD Dr. Werner Neundorf
Jahr der Mathematik 2008
Herzlich Willkommen an der TU Ilmenau
Zeige mal, was du kannst!
www.tu-ilmenau.de/site/math
Jubiläumskalender zum Jahr der Mathematik 2008
Körperzauber & Traumlandschaften

1.3 Von der Formel zur Zauberkiste

Im Rahmen der Lehre und Forschung spielt auch die Betrachtung von “kleineren“ Aufgaben und Randproblemen eine Rolle. In der studentischen Ausbildung in Fächern wie Grundlagen der Informatik, Algorithmen und Programmierung, Wissenschaftliches Rechnen und Numerische Mathematik wurden zahlreiche einfache Anwendungsfälle und akademische Beispiele erörtert, modelliert und in verschiedenen höheren Programmiersprachen wie BASIC, Pascal, PL/1, FORTRAN und C implementiert. Später sind systematisch auch die CAS mit ihren leistungsfähigen Tools einbezogen worden.

Dabei handelte es sich oft um Probleme, Begriffe, Eigenschaften, Darstellungen oder Verfahren wie

- Lösung von linearen Gleichungssystemen,
- Matrixeigenschaften, wie Norm, Kondition, Eigenwerte,
- korrekt gestellte Aufgaben,
- Subtraktionskatastrophe,
- Termumformungen, mathematische und numerische Äquivalenz,
- Fehlerfortpflanzung und Fehleranalyse, Stabilität in Algorithmen,
- Berechnung mathematischer Konstanten,
- Näherungszahlen und Stellenwertsysteme,
- interne Zahlendarstellung und Zahlenformate,
- Kurvendiskussion, Wachstumsmodelle, Mehrkörperprobleme,
- rekursive und iterative Algorithmen,
- Verarbeitung von umfangreichen Datenmengen,
- spezielle Computereffekte,
- geometrische Sachverhalte,
- grafische Animationen.

Zahlreiche Sachverhalte sind in Arbeiten wie den Preprints [3] - [12] dargestellt.

Im Rahmen der Öffentlichkeitsarbeit des Instituts für Mathematik habe ich darauf aufbauend zahlreiche Vorträge mit unterschiedlichem Anforderungsniveau konzipiert. Das Angebot wird ständig erweitert und aktualisiert.

Angebote für Schülervorträge

1. **Fehlerquellen in Rechnungen - Woher kommen diese und sind sie vermeidbar?**

   Überschaubare Beispiele und Demonstrationen mit Unterstützung durch kleine Computerprogramme und grafische Tools erweisen sich als wirksame und moderne didaktische Werkzeuge zur Illustration von

   - Fehlerbetrachtungen,
   - Termumformungen,
   - Kurvendiskussion

   und anderen Fragestellungen der angewandten Mathematik.

2. **Das Dilemma der alten Griechen: Besteht eine Figur aus Punkten?**
   - Was beinhaltet die “Faden“-Vorstellung bei der Transformation von Bereichen?
   - Können sich Kurven verschiedener Länge beliebig nahe kommen?
   - Wie hilft die “Faden“-Vorstellung bei der einfachen Berechnung der Kreisfläche?
   - Lässt sich das “Faden“-Konzept auf die Bestimmung der Kugeloberfläche übertragen?

3. **Das Phänomen π und e**

   Folgen und Reihen, Grenzwerte, zahlentheoretische Aspekte, Differential- und Integralrechnung, rekursive Beziehungen sowie geometrische Anschauungen werden gebraucht, um eine Vielzahl von Informationen über interessante mathematische Konstanten zu erhalten.
   Aus diesem Einblick öffnen wir nur ein kleines Fenster, wo wir uns mit ausgewählten Formeln und trickreichen Berechnungen den Zugang zur Kreiszahl und der Eulerschen Zahl verschaffen.

4. **Zeige mal, was du kannst!**
   - Dreieck mit größter Fläche
   - Dreieck mit kleinster Fläche
   - Entfernung des Schiffes vom Leuchtturm
   - Zuckerhut und Gondel
   - Feuerwehrleiter
   - Tischtennisbälle im Flugzeug
   - Anteil der gefärbten Fläche zur Gesamtfläche
   - Anzahl der Würfel im Bauwerk
   - Figur mit Kästchen
   - Verhältnis von Fläche zu Umfang
   - Besselscher Irrgarten
   - Differentialgleichung
   - Unbestimmtes Integral
   - Klassisches und modifiziertes Newton-Verfahren
   - Quadratwurzel mittels Höhensatz im Dreieck
   - Satz von Pythagoras
   - Kreis und rechtwinkeliges Dreieck
   - Graphics Showcase
   - $1 = 0$?
   - Minimaler Verbindungsweg
   - Uhr mit Zeiger und Uhrzeiger mit Bewegung
   - Tannenbäume mit Weihnachtskugel bzw. Osterei
   - Zahlenraten, Denkprobleme
   - Summe von Zahlen und Teilbarkeit

- Faktorisierung von Zahlen und Restklassenarithmetik
- Teilbarkeit, Restklassenarithmetik, Kongruenzen
- Schnelles Potenzieren
- Kryptographie - RSA-Public-Key-Kryptosystem
- Kryptographie - Textverschlüsselung
- Äquivalenz von Formeln
- Kreiszahl π
- Rekursion und Stabilität
- Torus
- Reihen und Konvergenz
- Das Gibbssche Phänomen
- Funktionen, Ableitungen, Grafen, Unstetigkeiten
- Seemannsknoten
- Rund um den Fußball
- Rettungsschwimmerproblem
- Archimedische Körper - Entfalten von archimedischen Koerpern Tetraeder, Hexaeder, Oktaeder, Ikosaeder, Dodekaeder
- Entfalten von Polyedern: kubisches Oktaeder
- Seemannsknoten: Trossenstek
- Seemannsknoten mit Animation

5. **Die Mathematische Zauberkiste**
   - Seemannsknoten
   - Rund um den Fußball
   - Rettungsschwimmerproblem
   - Körper und ihre Entfaltung
   - Wie rund sind der Kreis und die Kugel?
   - Spielwürfel und seine Konstruktion
   - Schachbrett und $1 = 0$?
   - Anzahl der Würfel im Bauwerk.

Spezielle Angebote für Vorträge in Spezialistenlagern mit Teilnehmern an Mathematik-Olympiaden

1. Interpolation und Splines
2. Zu Orthogonalsystemen von Polynomen und ihrer Rekursion
3. Lösung von Gleichungssytemen mittels Abstiegsverfahren
4. Das modifizierte Gradientenverfahren

Inzwischen hat sich die Zauberkiste weiter gefüllt.
Einige Themen wurden ausgebaut. Neues ist hinzugekommen. Weitere Anschauungsmodelle wurden konstruiert.

Die Mathematische Zauberkiste, Stand März 2010

Bisherige Themen + Erweiterungen
- Seemannsknoten mit Animation
- Wie rund sind Flächen und Körper? Bewegungsabläufe
- Regelmäßige Flächen und Körper
- Falten von Flächen, Körpern und Figuren
- 3D-Flächen: Torus, Möbiusband, Affensattel, Schnecke, Kugel, Ellipsoid, Würfel
- 2D-Flächen: Karo
- 2D-Kurven
- Schachbrett und Reiskörner
- Glas, Sektglas, Eisbecher
- Mandelbrot-Mengen, Julia-Mengen, Einzugsgebiete
- Magische Quadrate, Konstruktion und Animation
- Populationsmodell: Gras-Hasen-Füchse-Jäger
- Verschiedenes: Hilbert-Kurven, Lichterkette, Maple-Zeichen, ...

Im Angebot befinden sich auch etwas "abseits" liegende Themen wie Fotografieren durch Prismen, Fotomontage und 3D-Malerei, Puzzle-Spiele.

Heute findet man im Internet zu vielen Problemen dieser Art zahlreiche Hinweise, Software, Lösungswege, numerische Auswertungen und grafische Darstellungen einschließlich Animationen. Die Nutzung des Angebots ist jedoch die eine Seite, selbst mal etwas nachzuvollziehen schon schwieriger. Und inspiriert von diesen Ideen und Lösungen etwas Eigenes zu schaffen, kann durchaus recht mühsam sein. Dazu kommt dann immer noch eine möglichst perfekte Präsentation der Ergebnisse als Vortrag, als Demonstration oder in den Medien, das Internet eingeschlossen.

Trotzdem sollte man neugierig sein!

Ich gebe dazu nur einige interessante Internetadressen an.

```
www.tu-ilmenau.de/fakmn/Maple.3528.0.html
math.haifa.ac.il/ROVENSKI/rovenski-compgeom.html
kortenkamps.net/material/EulerTour/
kortenkamps.net/tiki-index.php?page=Projekte
www.cinderella.de
www.mathematikum.de
www-m10.ma.tum.de/ix-quadrat
www-history.mcs.st-and.ac.uk/Indexes/Hist_Topics_alph.html
www-history.mcs.st-and.ac.uk/HistTopics/Mathematical_games.html
www.seemannsknoten.info
www.klabautermann.de/knotentafel/
```

Die Beispiele weisen darauf hin, wie komplex die Verwendung des Computers als Werkzeug und Medium der Lernens und Lehrens allgemein und besonders in der Mathematikausbildung zu sehen ist. Aber der Komfort eines Computers einschließlich der Software kann sich nur "entfalten" durch den an Ideen reichen und kreativen Nutzer in der Einheit mit seiner gezielten Nutzung, gründlichen Analyse und ständigen Weiterentwicklung.

Kapitel 2

Ausgewählte Beispiele

Aus der Vielfalt des Angebots der Mathematischen Zauberkiste kann ich nur Stichproben geben. Die ausgewählten Beispiele werden mehr oder weniger ausführlich beschrieben. Das betrifft Rechnungen, Algorithmen, Software, Lösungen, Grafik u.a. Trotzdem hoffe ich, dass der Leser das Problem, die Herangehensweise und Lösungswege erkennt.

Einen einfachen Trick benutzt man beim Zahlenraten.

(1) Rate die Zahl, die sich ein anderer merkt.

Person A merkt sich die Zahl a. Person B rät diese, indem er A eine Aufgabe stellt.

```
Aufgabe=(2*a+9)*4-36,          B_raet=Aufgabe/8
Aufgabe=((2*(a-3)+6)*4-1)*2+2, B_raet=Aufgabe/16
```

(2) Eine Zahl wird in 2 Summanden zerlegt. Der erste Summand ist 50% des zweiten Summanden. Der zweite Summand ist um 50 größer als 50% des ersten Summanden. Um welche Zahl handelt es sich?

Aus den vielen Demonstrationen werden einige signifikante vorgestellt.
Typische numerische Beispiele findet man auch in [7]. In der Darstellung der Beispiele werden neben dem Thema und inhaltlichen Aspekten auch Bezug genommen auf Dateiverzeichnisse, Dateien, Demos, Material und Modelle.

2.1 Magische Quadrate

Verzeichnis: `magic`
Dateien: magic3,4,5,6,7,8.mw, magic5m.mw, magic7m.mw

- M(3,3), Trick der Erzeugung

1	2	3
4	5	6
7	8	9

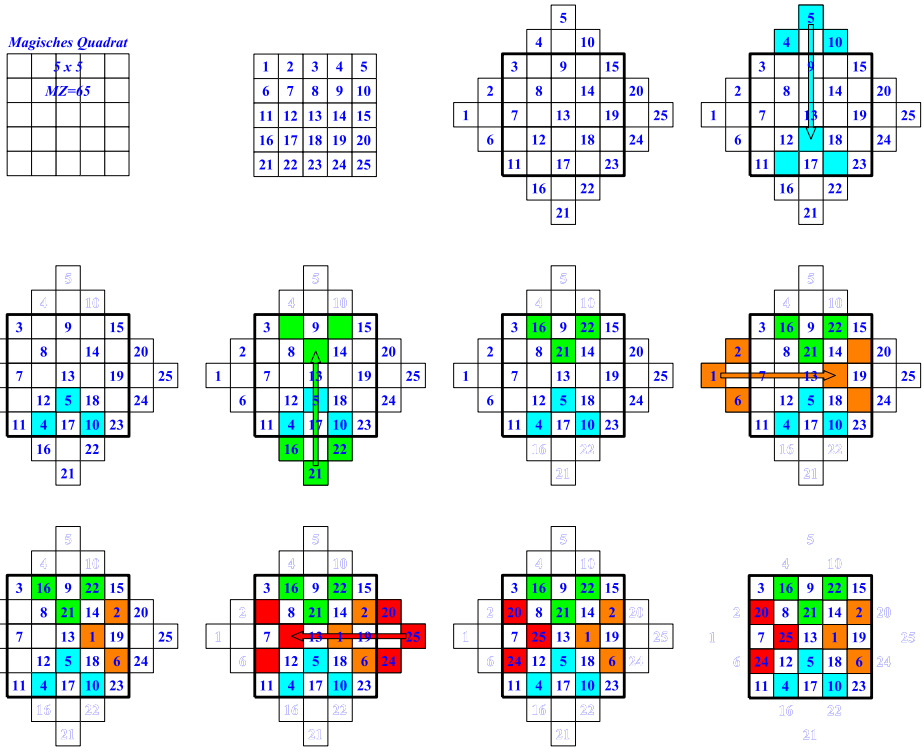
$\Rightarrow$

2	7	6
9	5	1
4	3	8

- M(5,5), PC-Demo/Animation mit Verallgemeinerung des Tricks sowie Lösungen

3	16	9	22	15
20	8	21	14	2
7	25	13	1	19
24	12	5	18	6
11	4	17	10	23

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9



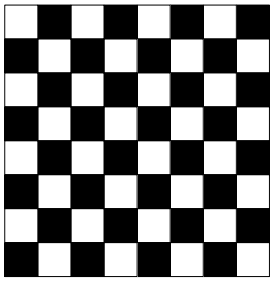
- Einfache Konstruktion für $M(n,n)$, n ungerade Dimension
- M(8,8), andere Methode der Erzeugung für n gerade
  als Übergang zum Schachbrett

```
M(8,8) =
        64  2  3 61 60  6  7 57
         9 55 54 12 13 51 50 16
        17 47 46 20 21 43 42 24
        40 26 27 37 36 30 31 33
        32 34 35 29 28 38 39 25
        41 23 22 44 45 19 18 48
        49 15 14 52 53 11 10 56
         8 58 59  5  4 62 63  1
```

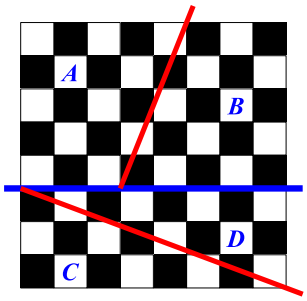
2.2 Numerisch-geometrische Beispiele

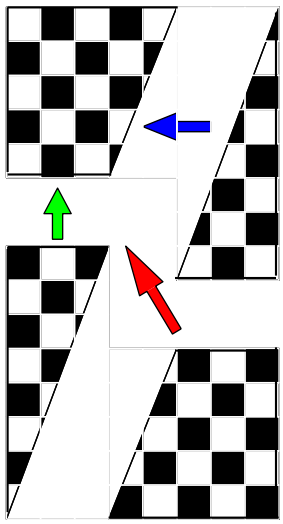
2.2.1 Schachbrett und $1 = 0$?

Verzeichnis: `schachbrett_1gleich0`
Dateien: schule5.pdf, schachbrettzerl1.mws, schachbrettzerl1.mw, Ani33_.mws
Modelle: Schachbrett, Schachbrett aus Papier



Dieses Karo aus $8 \times 8 = 64$ kleinen quadratischen Feldern wollen wir mal entgegen allen Gebräuchen zerschneiden.





Mit den 3 gut erkennbaren Schnitten zerfällt das Schachbrett in 4 Teilflächen **A**,**B**,**C** und **D**. Nun setzen wir dieselben Stücke nach kleiner Drehung wieder zusammen, aber etwas anders zu einem Rechteck. Wir berechnen die Rechteckfläche. Beide Flächen sind natürlich gleich.

$$F_1 = \mathbf{A}+\mathbf{B}+\mathbf{C}+\mathbf{D} = F_2 \;\Rightarrow\; 64 = 65 \;\Rightarrow\; 0 = 1$$

Zusatzaufgabe: Es gilt

$$1 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \ldots$$

Wie kann man damit $2 = \sum_{k=1}^{\infty} \frac{k}{2^k}$ zeigen?

2.2.2 Zwei Varianten für $1 = 0$?

Verzeichnis: `schachbrett_1gleich0`
Datei: schule5.pdf

(1) Es gilt

$$1 \;=\; \frac{1}{2}+\frac{1}{6}+\frac{1}{12}+\frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...,$$

auch wenn es mühsam ist, diese unendliche Summe nachzurechnen. Beachtet man

$$\frac{1}{n(n+1)} \;=\; \frac{1}{n}-\frac{1}{n+1},$$

so fällt einem der Nachweis leicht. Denn wir haben dann

$$\begin{aligned}
&\frac{1}{2}+\frac{1}{6}+\frac{1}{12}+\frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+... \\
&\quad=\Big(1-\frac{1}{2}\Big)\;+\Big(\frac{1}{2}-\frac{1}{3}\Big)\;+\Big(\frac{1}{3}-\frac{1}{4}\Big)\;+\Big(\frac{1}{4}-\frac{1}{5}\Big)\;+\Big(\frac{1}{5}-\frac{1}{6}\Big)\;+... \\
&\quad=1+\Big(-\frac{1}{2}+\frac{1}{2}\Big)+\Big(-\frac{1}{3}+\frac{1}{3}\Big)+\Big(-\frac{1}{4}+\frac{1}{4}\Big)+\Big(-\frac{1}{5}+\frac{1}{5}\Big)+\Big(-\frac{1}{6}+\frac{1}{6}\Big)+... \\
&\quad=1+0+0+0+....\;=\;1.
\end{aligned}$$

Wir notieren nun unsere Ausgangsgleichung

$$I: \quad \mathbf{1} \;=\; \frac{1}{2}+\frac{1}{6}+\frac{1}{12}+\frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...,$$

ziehen auf beiden Seiten $\frac{1}{2}$ ab, $1-\frac{1}{2}=\frac{1}{2}$ und bekommen

$$II: \quad \frac{1}{2} \;=\; \frac{1}{6}+\frac{1}{12}+\frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...,$$

ziehen auf beiden Seiten nun $\frac{1}{6}$ ab, $\frac{1}{2}-\frac{1}{6}=\frac{1}{3}$,

$$III: \quad \frac{1}{3} \;=\; \frac{1}{12}+\frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...,$$

ziehen auf beiden Seiten nun $\frac{1}{12}$ ab, $\frac{1}{3}-\frac{1}{12}=\frac{1}{4}$,

$$IV: \quad \frac{1}{4} \;=\; \frac{1}{20}+\frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...,$$

ziehen auf beiden Seiten nun $\frac{1}{20}$ ab, $\frac{1}{4}-\frac{1}{20}=\frac{1}{5}$,

$$V: \quad \frac{1}{5} \;=\; \frac{1}{30}+\frac{1}{42}+\frac{1}{56}+...$$

und so weiter.

Wir bilden die Summe S_1 aller linken Seiten der Gleichungen $I, II, III, IV, ...$, also $S_l = I_{links} + II_{links} + III_{links} + ...$, genauso die Summe S_r aller rechten Seiten. Es gilt $S_l \;=\; S_r$.

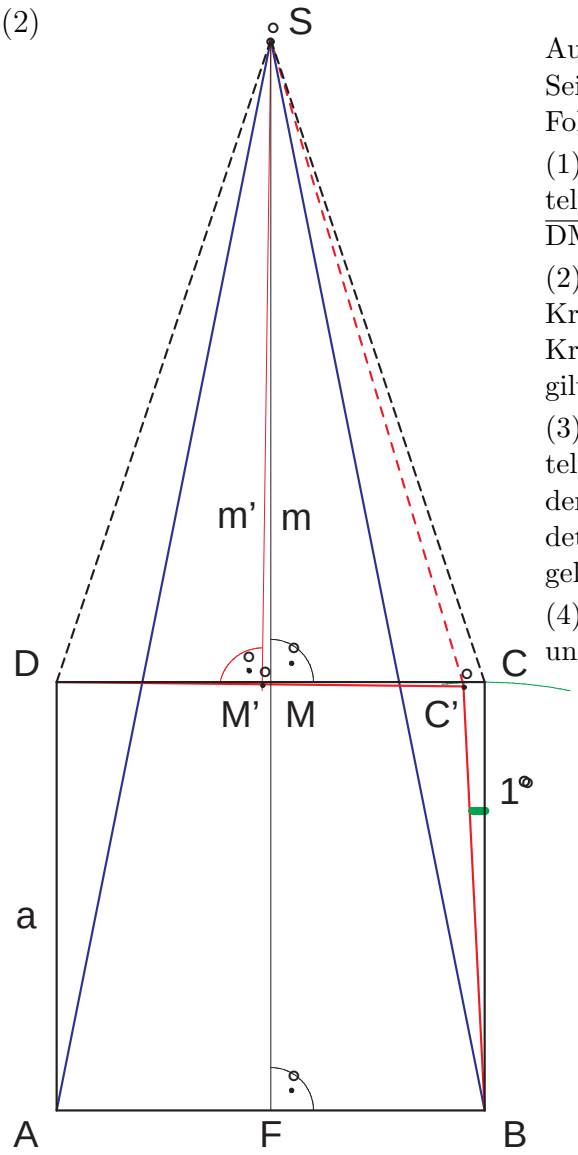
Bevor wir diese Summen ausrechnen, kürzen wir. Denn wir wissen, dass man gleiche Glieder auf beiden Seiten von Gleichungen wegkürzen kann.

links	rechts
$\frac{1}{2}$ in II_{links} mit	$\frac{1}{2}$ in I_{rechts}
$\frac{1}{3}$ in III_{links} mit	$\frac{1}{6}+\frac{1}{6}=\frac{1}{3}$ in I_{rechts}, II_{rechts}
$\frac{1}{4}$ in IV_{links} mit	$\frac{1}{12}+\frac{1}{12}+\frac{1}{12}=\frac{1}{4}$ in $I_{rechts}, II_{rechts}, III_{rechts}$
$\frac{1}{5}$ in V_{links} mit	$\frac{1}{20}+\frac{1}{20}+\frac{1}{20}+\frac{1}{20}=\frac{1}{5}$ in $I_{rechts}, II_{rechts}, III_{rechts}, IV_{rechts}$

und immer so weiter.

Was bleibt nach dem Wegstreichen auf beiden Seiten stehen? $S_l = S_r$ bzw. $1 = 0$.

(2)



Qualitative Darstellung der Situation

Ausgangspunkt ist ein Quadrat ABCD mit der Seitenlänge a.
Folgende Schritte werden gemacht:

(1) Auf der Seite CD errichtet man im Mittelpunkt M die Mittelsenkrechte m, damit ist $\overline{DM} = \overline{MC} = \frac{a}{2}$ und m $\perp$ CD, m $\perp$ AB.

(2) Um den Eckpunkt B schlägt man einen Kreis mit dem Radius a. Man zeichnet den Kreissektor C'BC mit dem Innenwinkel 1^o. Es gilt natürlich $\overline{BC'} = \overline{BC}$=a.

(3) Auf der Strecke C'D errichtet man im Mittelpunkt M' die Mittelsenkrechte m'. Auf Grund der leichten Neigung von m' nach rechts, schneidet sie die Mittelsenkrechte m im Punkt S. Es gelten $\overline{DM'} = \overline{M'C'}$ und m' $\perp$ C'D.

(4) Wir zeichnen die Strecken DS, CS, C'S, AS und BS ein.

Es gelten die folgenden Beziehungen in den verschiedenen Dreiecken:

ΔDM'S $\cong$ ΔC'M'S $\Rightarrow$ $\overline{DS} = \overline{C'S}$

ΔAFS $\cong$ ΔBFS $\Rightarrow$ $\overline{AS} = \overline{BS}$

ΔASD $\cong$ ΔBSC' weil *SSS*, d. h.
$\overline{AD} = \overline{BC'}$=a, $\overline{DS} = \overline{C'S}$, $\overline{AS} = \overline{BS}$

Damit gilt $\angle$ DAS = $\angle$ C'BS.

Weiterhin sind $\angle$ SAF = $\angle$ SBF und

$$\begin{array}{ccc} \angle \text{ DAS} + \angle \text{ SAF} & = & 90^o \\ \| & & \| \\ \angle \text{ C'BS} + \angle \text{ SBF} & = & 89^o \end{array}$$

$90 = 89$ bzw. $1 = 0$.

2.2.3 Schachbrett und Körner

Verzeichnis: `schachbrett_1gleich0`
Datei: schachbrett_koerner.txt
Modelle: Schachbrett, Beutel Reis ($125g$)

Dazu die Gschichte bzw. Legende
Sie stammt aus Persien und ist fast 800 Jahre alt.
Vor langer Zeit regierte in Indien der Herrscher Shihram. Er war ein Tyrann und seine Untertanen litten sehr unter ihm. Da erfand der Weise Sissa das Schachspiel. Mit diesem Spiel wollte er dem strengen Herrscher zeigen, wie wichtig für einen König seine Untertanen sind. Der König auf dem Schachbrett braucht die Bauern, Läufer, Springer und so weiter - ohne sie ist er verloren. Und genauso ist ein wirklicher König auf seine Untertanen angewiesen. Das sollte der strenge indische König lernen. König Shihram verstand diese Belehrung gut. Das neue Spiel gefiel ihm sehr gut und so wurde er ein begeisterter Schachspieler. Er befahl, dass das Schachspiel im ganzen Land verbreitet werden soll.
König Shihram war dem Weisen Sissa sehr dankbar für das neue Spiel und für die Belehrung. Er führte Sissa zu seiner Schatzkammer und sagte ihm: "Du darfst dir wünschen was du willst, du sollst es bekommen!" Sissa dachte nach und sagte dann zum König: "Ich wünsche mir nichts von deinen Schätzen. Ich habe einen anderen Wunsch."
Er ging mit dem König zu einem Schachbrett und sagte dann zu ihm: "Das ist mein Wunsch: Ich möchte Weizenkörner von dir. Lege auf das erste Feld des Schachbretts ein Korn und dann auf jedes weitere Feld des Schachbretts doppelt so viele Körner wie auf dem Feld davor."
Da wurde der König zornig. Er schrie: "Ich habe dir all meine Schätze angeboten, und du willst nur ein paar Weizenkörner von mir haben? Willst du mich beleidigen?" "O nein, mein Herr", sagte Sissa, "bestimmt möchte ich dich nicht beleidigen. Bitte erfülle mir meinen Wunsch, dann wirst du sehen, dass es ein großer Wunsch ist."
Der König rief seine Diener und befahl ihnen, das Schachbrett so mit Körnern zu belegen, wie Sissa es wünschte. Die Diener holten Weizen und fingen damit an. Doch schon bald merkten sie: Es ist unmöglich, diesen Wunsch zu erfüllen. Sie kamen zum König und sagten zu ihm: "Wir können Sissas Wunsch nicht erfüllen." "Warum nicht?" fragte der König wütend. Da antworteten sie ihm: "Aller Weizen unseres Landes und dazu der Weizen unserer Nachbarländer ist nicht genug, um diesen Wunsch zu erfüllen. So viel Weizen gibt es gar nicht."
So hat der König Shihram eine zweite Belehrung von Sissa erhalten. Er hat gelernt, dass man das Kleine und Geringe nicht unterschätzen soll.

Nun zum Modell.
Schachbrett, Weizenkörner/Reiskörner,
Beutel Reis ($125\ g$),
1 Korn=$0.025\,g = \frac{1}{40}\,g \ \Rightarrow$ 5000 Körner im Beutel

Dauer des Einsammelns der Reiskörner bei einer Dauer von 1sec/Korn?
Belegung des Schachbretts mit Reiskörnern wie folgt

1	2	4	8	16	32	64	128
256	512	1024	2048	4096	9192	...	...
...							
						...	L

$L = 9\,223\,372\,036\,854\,775\,808 \approx 9$ Trillionen $= 9 \cdot 10^{18}$

1 LKW $\sim$ $10\,t$ Lademasse $\sim$ $10\,m$ lang

Wie lang ist die LKW-Schange, die für den Abtransport der Reiskörner auf dem Schachbrett gebraucht wird? Reicht die Schlange einmal um die Erde oder von der Erde zum Mond oder vielleicht von der Erde bis zur Sonne?

Verwendung der Astronomischen Einheit = AE $\approx$ 150 Millionen km

2.2.4 Kreis und rechtwinkeliges Dreieck

Verzeichnis: `kreis_und_dreieck`
Dateien: kreis1.mws, zeige_mal2.mws
Modell: Teppichkreis

Die Fläche des Einheitskreises $F_\circ = \pi r^2$, $r = 1$, lässt sich transformieren auf die Fläche eines rechtwinkeligen Dreiecks $F_\triangle = \frac{1}{2} r \cdot 2\pi r,\ r = 1$.

Die Darstellungen des Sachverhalts beruht auf flächendeckenden Kurven (Mäander). Man stelle sich den Einheitskreis vor überdeckt von vielen konzentrischen Kreisen in Form von "dicken Wollfäden". Diese sind an ihrem Anfang auf der Strecke $x = [0, 1]$ festgemacht und das andere Ende ist frei. Durch eine vertikale Luftströmung von unten kommend richten sich die Fäden auf und bilden flächendeckend ein rechtwinkeliges Dreieck.

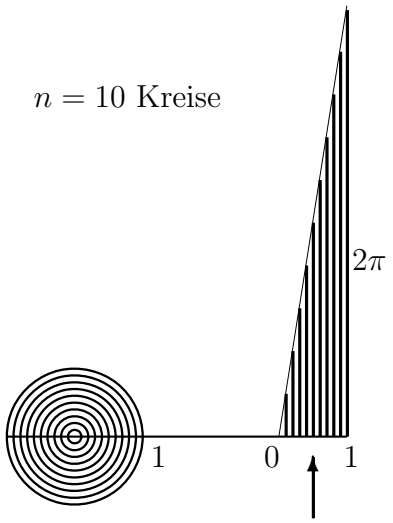


Abb. 2.1 Einheitkreis
= rechtwinkeliges Dreieck

Nun wollen wir die Vorgehensweise zur Darstellung der Flächenübereinstimmung in Maple implementieren, wobei wir mit einfachen Figuren beginnen und bei der Animation für das "Aufrichten" der Fäden enden werden.

Rechnungen in Maple

Formel der Kreisfläche $F_\circ = \pi r^2$ aus Umfang und rechtwinkeligem Dreieck
Kreisgleichung, Einheitskreis

```
> r:=1:
  u:=[r*cos(x),r*sin(x),x=0..2*Pi]:
  plot(u,thickness=3,color=black,scaling=constrained);
```

$n = 10$ Kreise (flächendeckend, Mäanderisierung)

```
> n:=10:
  u:=r->[r*cos(x),r*sin(x),x=0..2*Pi]:
  plot([seq(u(i/n),i=1..n)],thickness=15,scaling=constrained,
        color=[seq(COLOR(RGB,2*i/n*rand()/10^12,
                             2*i/n*rand()/10^12,
                             2*i/n*rand()/10^12),i=1..n)]);
> pl1:=plot([seq(u(i/n),i=1..n)],thickness=15,scaling=constrained,
              color=[seq(COLOR(RGB,2*i/n*rand()/10^12,
                                   2*i/n*rand()/10^12,
                                   2*i/n*rand()/10^12),i=1..n)]):
```

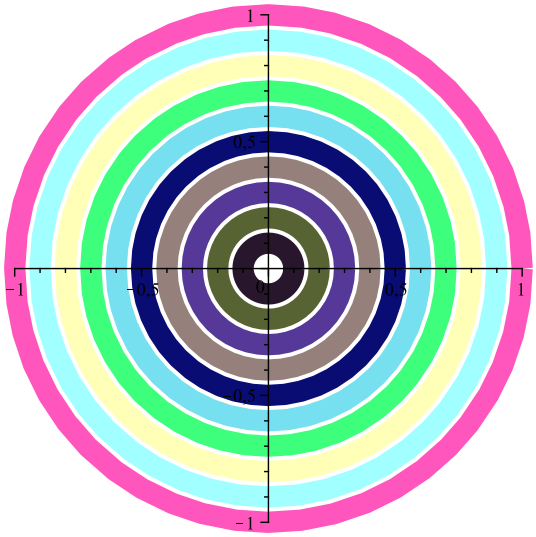


Abb. 2.2 $n = 10$ Kreise (flächendeckend, Mäanderisierung)

Animation: Aufrichten des Einheitskreises zu einer vertikalen Strecke

```
> animate([(1-t)*cos(x)+t,(1-t)*sin(x)+x*t,x=0..2*Pi],t=0..1,
           scaling=constrained,frames=16,color=gold,thickness=12);
> plots[display](op(1, op(1,%)),scaling=constrained,
        thickness=12,view=[-1..1,-1..6.3]);    # 1. Frame
  plots[display](op(16,op(1,%%)),scaling=constrained,
        thickness=12,view=[-1..1,-1..6.3]);    # letztes Frame
```

Einheitskreis ($t = 0$) bis vertikale Strecke der Länge 2π bei $0 \leq x \leq 2\pi$

```
> v:=(t,x)->(1-t)*cos(x)+t:
  w:=(t,x)->(1-t)*sin(x)+x*t:
```

Bildfolge von Kreis ($t = 0$) mit Radius r bis vertikale Strecke der Länge $2\pi r$ für alle n Kreise

```
> m:=15:   # m+1 = Anzahl der Frames
  p1:=(k,j)->plot([k/n*v(j/m,x),k/n*w(j/m,x),x=0..2*Pi], # r=k/n, k=1..n
              scaling=constrained,thickness=4,
              color=COLOR(RGB,(m-j+1)/(m+3),(j+1)/(m+3),(m-j+1)/(m+3))):
```

Äußerer Kreis (Nummer n)

```
> p1(n,1);     # plots[display](p1(n,1));   # 2. Frame nach Kreis

> p2:=[seq(p1(n,j),j=0..m)]:
  # Animation
  plots[display](p2,insequence=true,view=[-1..1,-1..6.3],
                 scaling=constrained);
```

Bildfolge für den äußeren Kreis (16 Frames) mit Darstellung
1. Version

```
> m2:=(m+1)/2:
  ph:=array(1..m+1,[]): pp:=array(1..2,1..m2,[]):
  for l from 0 to m do
   ph[l+1]:=display(p1(n,l)):
  end do:
  for l from 1 to m2 do
   pp[1,l]:=display(ph[l],tickmarks=[0,0]):
   pp[2,l]:=display(ph[m2+l],tickmarks=[0,0]):
  end do:
  plots[display](ph);   # 16 Bilder nebeneinander, zu eng
  plots[display](pp);   # 16 Bilder als Tableau 2*8, guenstiger
```

Abb. 2.3 Bildfolge für den äußeren Kreis ($1 * 16$ Frames),
ungünstige Darstellung, wird durch Skalierung des *ps*-Files nicht besser

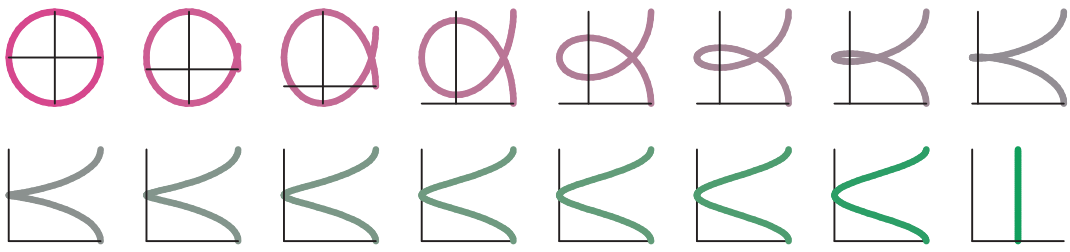


Abb. 2.4 Bildfolge für den äußeren Kreis ($2 * 8$ Frames)

2. Version

```
> p1m:=(k,j)->plot([k/n*v(j/m,x),k/n*w(j/m,x),x=0..2*Pi], # [1,6.3] },
              scaling=constrained,thickness=4,tickmarks=[0,0],
              view=[-3.5..3.5,-1..6.3],
              color=COLOR(RGB,(m-j+1)/(m+3),(j+1)/(m+3),(m-j+1)/(m+3))):
  pz:=plot([[3.5,-1],[3.5,6.3],[-3.5,6.3]],color=white):
  # Zwangsmassnahme
  p1mm:=(k,j)->display([p1m(k,j),pz],scaling=constrained):
    display(p1mm(n,0));
  display(p1mm(n,m));
> m4:=(m+1)/4:
  pp:=array(1..4,1..m4,[]):
  for l from 1 to m4 do
   pp[1,l]:=display(p1m(n,l-1)):
   pp[2,l]:=display(p1m(n,m4+l-1)):
   pp[3,l]:=display(p1m(n,2*m4+l-1)):
   pp[4,l]:=display(p1m(n,3*m4+l-1)):
  end do:
  plots[display](pp);   # 4*4 Bilder nebeneinander, nicht skaliert
> for l from 1 to m4 do
   pp[1,l]:=display(p1mm(n,l-1)):
   pp[2,l]:=display(p1mm(n,m4+l-1)):
   pp[3,l]:=display(p1mm(n,2*m4+l-1)):
   pp[4,l]:=display(p1mm(n,3*m4+l-1)):
  end do:
  plots[display](pp);   # 4*4 Bilder nebeneinander, skaliert
```

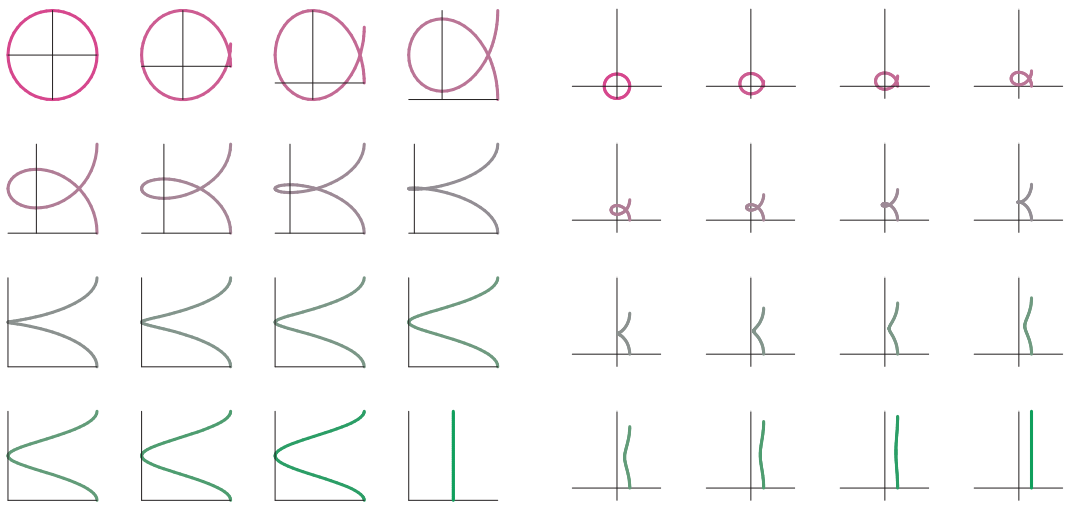


Abb. 2.5 Bildfolge für den äußeren Kreis ($4 * 4$ Frames),
links: nicht skaliert, rechts: skaliert

1. Frame für alle n Kreise

```
> p1(1,1);        # plots[display](p1(1,1));
> p3:=[seq(p1(k,1),k=1..n)]:
  # Animation
  plots[display](p3,insequence=true,thickness=4,scaling=constrained);
```

Bildserie des 1. Frames für alle 10 Kreise

```
> pz:=plot([[1,1],[-1,1],[-1,-1]],color=white):
  ph:=array(1..n,[]):
  for l from 1 to n do
   ph[l]:=display([p1(l,1),pz],view=[-1..1,-1..1],tickmarks=[0,0]):
  end do:
  plots[display](ph);   # n=10 Bilder nebeneinander
```

Abb. 2.6 1. Frame für alle 10 Kreise

Bild mit allen Kreisen und Frames

```
> ph:=array(1..m+1,[]): pp:=array(1..4,1..m4,[]):
  for l from 0 to m do
   ph[l+1]:=display(seq(p1(k,l),k=1..n)):
  end do:
  for l from 1 to m4 do
   pp[1,l]:=display(ph[l],tickmarks=[0,0]):
   pp[2,l]:=display(ph[m4+l],tickmarks=[0,0]):
   pp[3,l]:=display(ph[2*m4+l],tickmarks=[0,0]):
   pp[4,l]:=display(ph[3*m4+l],tickmarks=[0,0]):
  end do:
> p4:=[seq(seq(p1(k,j),k=1..n),j=0..m)]:
  # Animation, 1.-16. Frame ueber jeweils alle 10 Kreise
  display(p4,view=[-1..1,-1..6.3],insequence=true,scaling=constrained);
> plots[display](ph);   # 16 Bilder nebeneinander
> plots[display](pp);   # 16 Bilder als Tableau 4*4
```

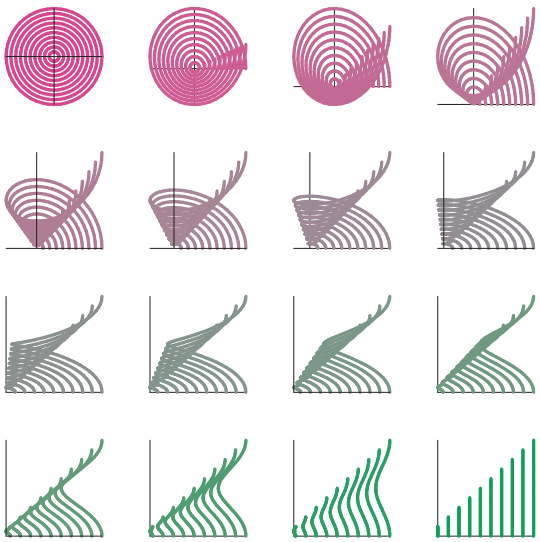


Abb. 2.7 Alle 16 Frames für alle 10 Kreise

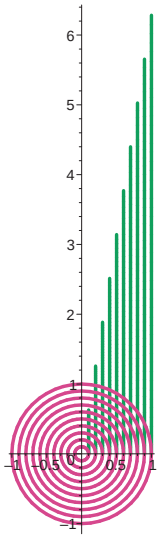


Abb. 2.8 1. und 16. Frame

Animation der “Aufspreizung“, Fäden etwas enger

```
> p11:=(k,j)->plot([k/n*v(j/m,x),k/n*w(j/m,x),x=0..2*Pi],
              scaling=constrained,thickness=6,
              color=COLOR(RGB,(m-j+1)/(m+3),(j+1)/(m+3),(m-j+1)/(m+3))):
  ph1:=array(1..m+1,[]):
  for l from 0 to m do
   ph1[l+1]:=display(seq(p11(k,l),k=1..n)):
  end do:
> picts:=[seq(ph1[l],l=1..m+1)]:
  display(picts,insequence=true,scaling=constrained,
          view=[-1..1,-1..6.3],title=‘F=r*(2*r*Pi)/2=Pi*r^2‘);
> # siehe Abb. ... : 1. Frame (alle Kreise) und
  # 16. Frame (alle Vertikalen, rechtwinkeliges Dreieck)
  display(ph1[1],ph1[m+1]);
```

2.2.5 Quadratwurzel mittels Höhensatz im Dreieck

Verzeichnis: `verschiedenes`
Dateien: hoehensatz.mws, zeige_mal2.mws

Der Höhensatz von Euklid im rechtwinkeligem Dreieck sagt, dass das Quadrat über der Höhe gleich dem Produkt der beiden anliegenden Hypotenuseabschnitte ist: $h^2 = pq$.

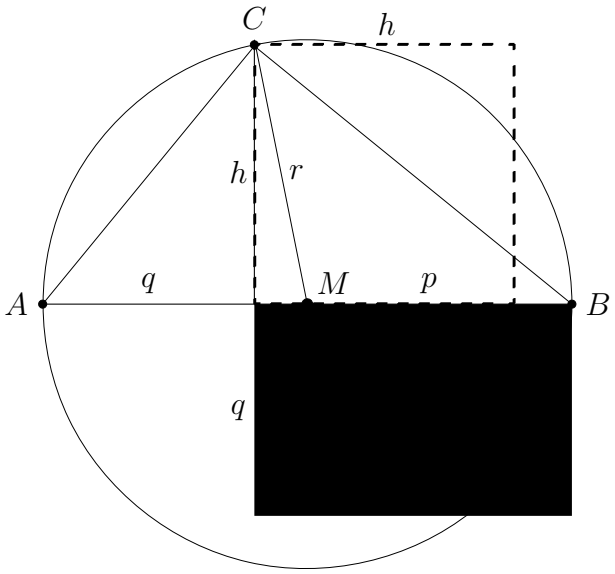


Abb. 2.9
Datei *hoehe*1.*pic*,
Höhensatz von Euklid $h^2 = pq$

Ähnlich kann man den Kathetensatz von **Euklid** oder den Satz von **Pythagoras** darstellen.
Wir wollen aber den Höhensatz zur Quadratwurzelberechnung aus einer gegebenen Zahl $t > 0$ verwenden. Dazu setzen wir im Höhensatz $t = p$ und $q = 1$ und erhalten somit $h = \sqrt{t}$, was sich nun in Maple schön demonstrieren lässt.

Rechnungen in Maple

```
> # Radikand
  t:=5;
  'sqrt(t)'=sqrt(t);
  'sqrt(t)'=sqrt(5.0);
> # Teilplots
  pl1:=plot([[0,0.01],[t,0.01]],-1..t,thickness=5,color=blue):
  pl2:=plot([[-1,0.01],[0,0.01]],-1..t,thickness=5,color=green):
  x0:=(t-1)/2:
  r:=(t+1)/2:
  h:=sqrt(r^2-(0-x0)^2):
  pl3:=plot(sqrt(r^2-(x-x0)^2),x=-1..t,thickness=2,color=red):
  pl4:=plot([[-1,0],[0,h],[t,0]],thickness=1,color=red):
  pl5:=plot([[0,0],[0,h]],thickness=5,color=red):
  pl6:=plot([[x0,-0.05],[x0,0.2]],thickness=2,color=blue):
  pl7:=plot([[x0,0]],style=point,symbol=solidcircle,
                     symbolsize=16,color=blue):
  pl8:=plot([[x0,0],[0,h]],color=brown):

> st:=convert(t,string):
  hk:=sqrt(r^2-(x-x0)^2):
  shk:=convert(hk,string):
  shk:=cat(`f(x)=sqrt(9-(x-2)^2)=`,shk):
  p1:=textplot([[0.8,1,` h=f(0)=sqrt(5)`],
                [2.5,0.3,` t=`||st],[3,3.2,shk]],font=[TIMES,ITALIC,11]):

> pp:=display([pl3,pl4,pl6,pl7,pl8,p1,pl5,pl1,pl2],scaling=unconstrained,
        view=[-1..5,-0.5..3.5],
        title=`    Hoehensatz im rechtwinkeligen Dreieck  h^2=1*t\n
                   -->  Wurzel aus t`,
        titlefont=[HELVETICA,BOLD,10]):
  display(pp);
```

$$t := 5$$

$$\sqrt{t} = \sqrt{5}$$

$$\sqrt{t} = 2.236067977$$

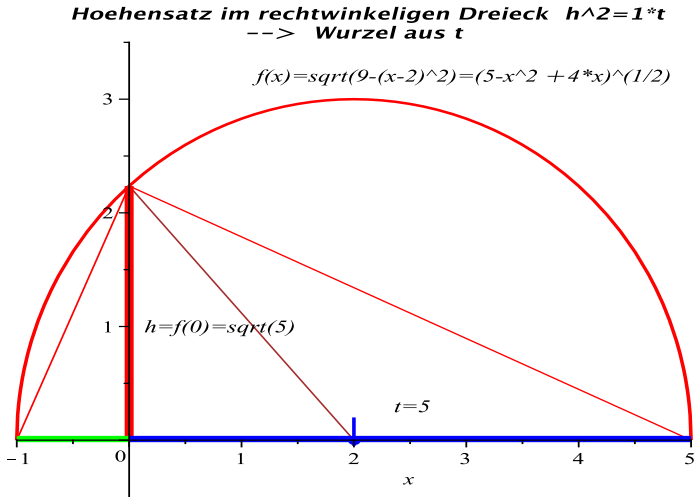
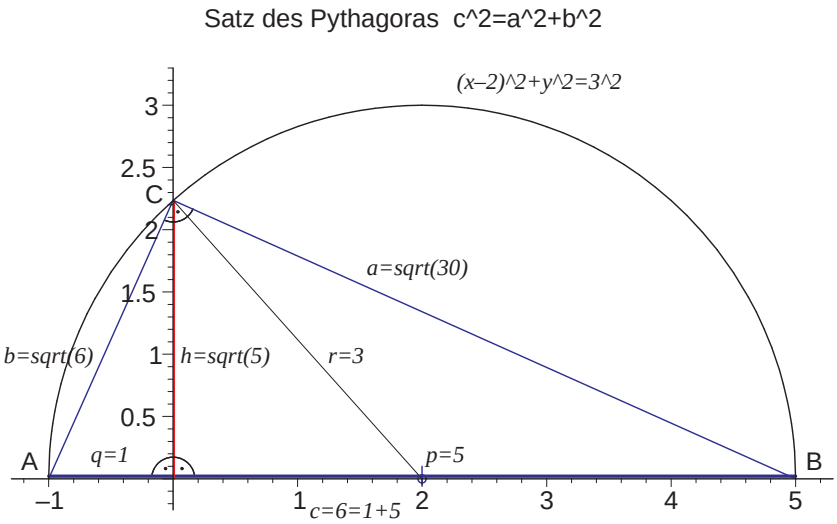


Abb. 2.10
Quadratwurzel $h = \sqrt{t},\ t > 0$, mittels Höhensatz

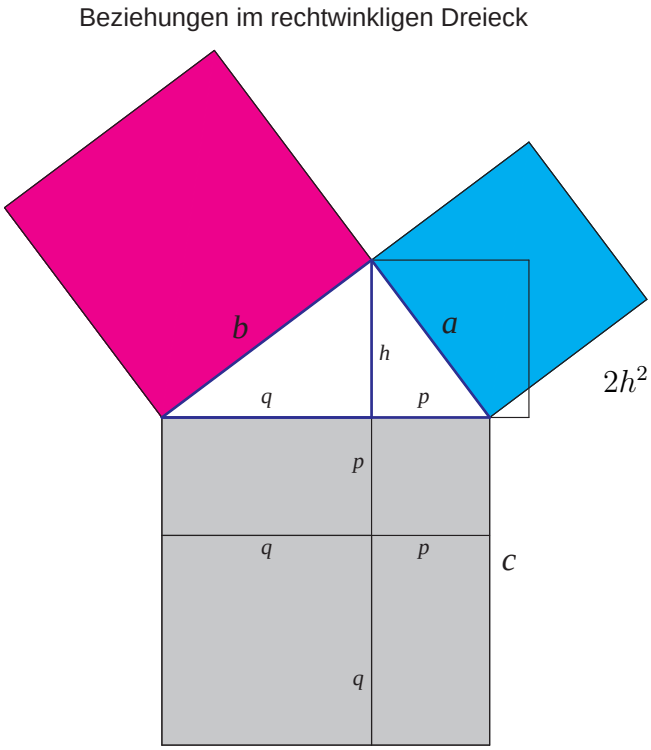
2.2.6 Satz des Pythagoras

Verzeichnis: `verschiedenes`
Dateien: pythagoras.mws, zeige_mal2.mws
Modell: Schnittmodell

Der Satz des Pythagoras von Samos (ca. 570-510 v.u.Z.) lautet $c^2 = a^2 + b^2$.

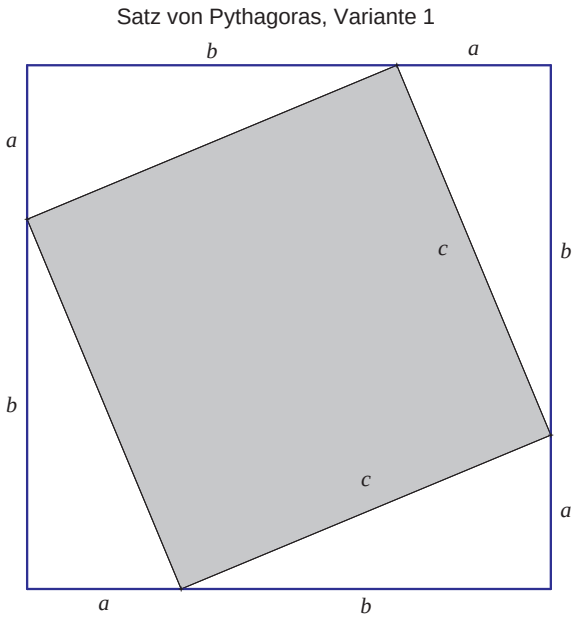


Es gelten im rechtwinkligen Dreieck die folgenden Beziehungen.



$$
\begin{aligned}
a^2 + b^2 &= c^2 \\
&= (p+q)^2 \\
&= p^2 + q^2 + 2pq \\
h^2 + p^2 &= b^2 \\
h^2 + q^2 &= a^2 \qquad + \\
2h^2 + p^2 + q^2 &= a^2 + b^2 = c^2 \\
&= p^2 + q^2 + 2pq \\
h^2 &= pq \quad \text{(Höhensatz)} \\
a^2 + b^2 &= c^2 \\
&= (p+q)c = pc + qc \\
a^2 &= pc \quad \text{(Euklid)} \\
b^2 &= qc
\end{aligned}
$$

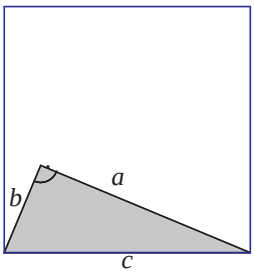
Variante 1 (mit Rechnung)

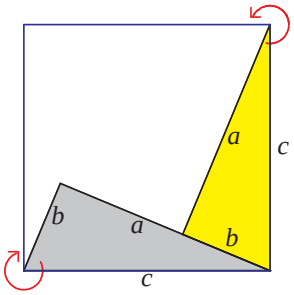


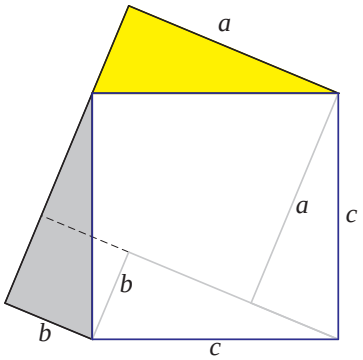
großes Quadrat
= kleines Quadrat + 4 Dreiecke

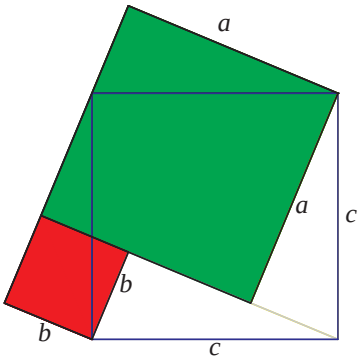
$$
\begin{aligned}
(a+b)^2 &= c^2 + 4\frac{ab}{2} \\
a^2 + b^2 + 2ab &= c^2 + 2ab \\
a^2 + b^2 &= c^2
\end{aligned}
$$

Variante 2 (nur grafisch)



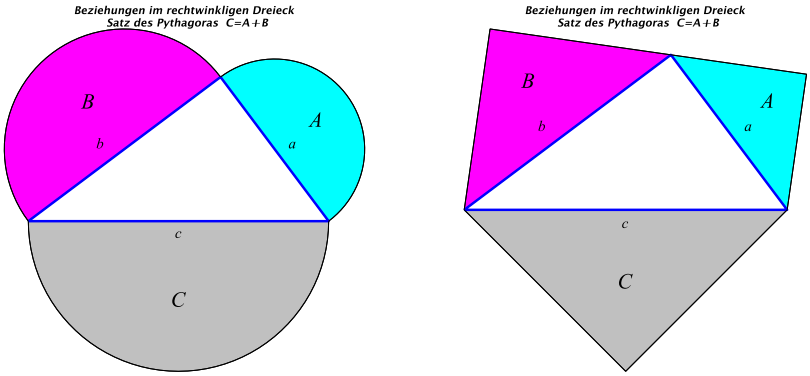






Weniger bekannt dürfte der Satz von Pythagoras **C=A+B** unter Verwendung von Halbkreisen bzw. gleichschenkeligen rechtwinkeligen Dreiecken sein.

Variante 3



2.2.7 Uhr mit Zeiger und Uhrzeiger mit Bewegung

Verzeichnis: `uhr_mit_zeiger`
Dateien: zeige_mal2.mws, watch2,3,4.m

Das Ziffernblatt einer Uhr mit Sekundenzeiger sei die Ausgangssituation für die folgende einfache Betrachtung. Wir wollen die 60 Positionen des Sekundenzeigers als Animation in Maple darstellen.

Rechnungen in Maple

Uhr mit Sekundenzeigerstellungen

```
> kreis:=plot([sin(t),cos(t),t=0..2*Pi],-1..1,-1..1,
                scaling=constrained,axes=none,thickness=2,color=black):
  zeiger1:=t->plot([[0,0],[sin(t),cos(t)]],thickness=3,tickmarks=[0,0]):
  n:=60:
  drehz1:=[seq(zeiger1(2*Pi*i/n),i=0..n)]:
  # Animation
  pdreh1:=plots[display](drehz1,insequence=true,scaling=constrained):
  plots[display](kreis,pdreh1);
```

Alle Zeigerstellungen

```
> plots[display](kreis,drehz1);
```

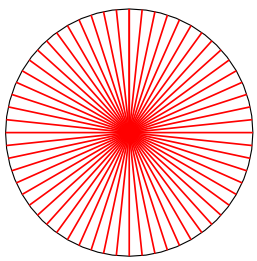


Abb. 2.11
Blatt einer Uhr
mit 60 Sekundenzeigerstellungen

```
> t:='t':
  kreism:=plot([0.05*sin(t),0.05*cos(t),t=0..2*Pi],-1..1,-1..1,
                axes=none,color=black,thickness=15):
  zeiger2:=t->plottools[arrow]([0,0],[0.8*sin(t),0.8*cos(t)],
                               0.1,0.2,0.2,color=red):
 # ev. noch Ziffernblatt

> n:=60:
  drehz2:=[seq(zeiger2(2*Pi*i/n),i=0..n)]:
  # Animation
  pdreh2:=display(drehz2,insequence=true,scaling=constrained):
> display(kreis,kreism,pdreh2);
> display(kreis,drehz2);
```

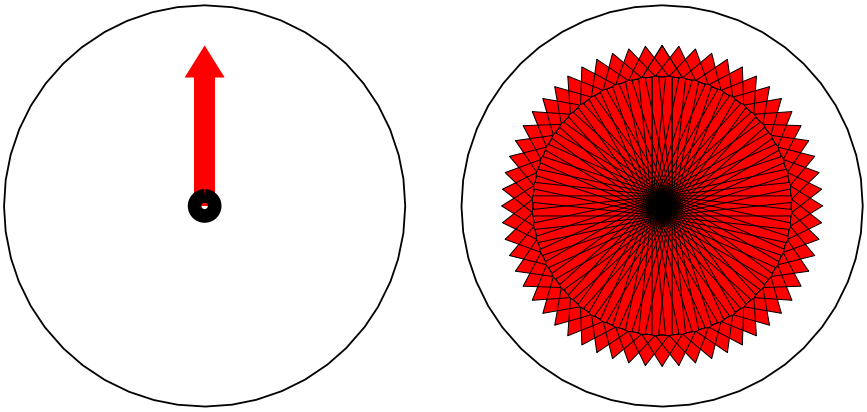


Abb. 2.12 Uhr mit Zeiger sowie mit 60 Sekundenzeigerstellungen

```
> n:=60:
  p:=polarplot([1,0.1,0.01],color=blue,thickness=[3,2,5]):
  ziffern:=textplot([seq([evalf(0.90*sin(Pi*i/6)),
                          evalf(0.90*cos(Pi*i/6)),i],i=1..12)],
                    font=[HELVETICA,BOLD,14]):
  q:=k->arrow([0,0],[evalf(0.80*sin(Pi*k/30)),evalf(0.80*cos(Pi*k/30))],
              0.05,0.15,0.15,color=green):
  uhr:=[seq(display([p,ziffern,q(k)]),k=0..n-1)]:
  display(uhr,insequence=true,axes=none,scaling=constrained);
```

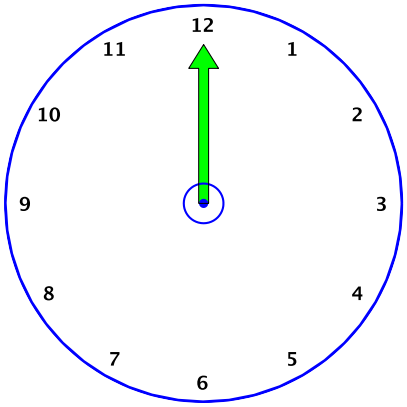


Abb. 2.13
Uhr mit Ziffernblatt sowie
mit Sekundenzeigerstellung

Nun soll auch in MATLAB die Uhr simuliert werden.

```
% watch4.m
% Simulation einer Analoguhr mit Ziffernblatt und Zeigerbewegung
% Funktion clock liefert die aktuelle Zeit als Feld
% clock = [year month day hour minute second]
clc
clear all
clf
figure(1)
hold on
fill([-3,3,3,-3],[-3,-3,3,3],'w'), axis 'off', axis 'equal'
t=0:0.02:2*pi;
plot(2.8*cos(t),2.8*sin(t),'k','LineWidth',3)
% polar(2*pi,3), axis 'off', axis 'equal'
for n=0:11
   theta=n*2*pi/12+pi/2;
   polar([theta theta],[2.2 2.4],'m')
   polar([theta-0.005 theta-0.005],[2.2 2.4],'m')
   polar([theta+0.005 theta+0.005],[2.2 2.4],'m')
   [x,y]=pol2cart(theta,2.6);
   text(x-0.05,y,int2str(12-n))
end
drawnow
t      =fix(clock);
houra  =rem(t(4),12);
minutea=t(5);
seconda=t(6);
angle1a=(12-houra)*2*pi/12+pi/2-(minutea/60)*2*pi/12;
angle2a=(60-minutea)*2*pi/60+pi/2;
angle3a=(60-seconda)*2*pi/60+pi/2;
secondd=0;

while 1
t     =fix(clock);
hour  =rem(t(4),12);
minute=t(5);
second=t(6);
angle1=(12-hour)*2*pi/12+pi/2-(minute/60)*2*pi/12;
angle2=(60-minute)*2*pi/60+pi/2;
angle3=(60-second)*2*pi/60+pi/2;
polar([angle1a angle1a],[0 1.7],'w')
polar([angle1 angle1],[0 1.7],'b')
drawnow
angle1a=angle1;
polar([angle2a angle2a],[0 2.1],'w')
polar([angle2 angle2],[0 2.1],'b')
drawnow
angle2a=angle2;
if abs(second-seconda-1)<0.05
  secondd=secondd+1;
  seconda=second;
```

```
  if seconda==59, seconda=-1; end
  polar([angle3a angle3a],[0 2.2],'w')
  polar([angle3 angle3],[0 2.2],'r')
  drawnow
  angle3a=angle3;
end
if secondd>30, break; end
end
hold off
```

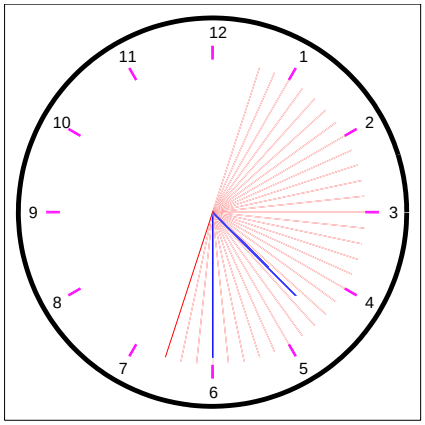


Abb. 2.14
Uhr mit Ziffernblatt, Simulation der Uhrzeit mit Zeigerbewegung (Laufzeit = 30 *sec*)

2.2.8 Tannenbäume mit Weihnachtskugel bzw. Osterei

Verzeichnis: `verschiedenes`
Dateien: weihnachten.mws, zeige_mal2.mws

Um eine Fichte oder Tanne relativ abstrakt darzustellen, benötigt man die Sägezahnkurve, die mit vertikaler und/oder horizontaler Stauchung/Streckung bearbeitet sowie mit ihren gespiegelten Teilen verbunden wird. Dazu bieten sich die mathematischen Standardfunktionen `floor, ceil, frac, trunc, round` an. Wir verwenden die Funktion, die von einer Zahl x die größte ganze Zahl $z \leq x$ bestimmt, also `floor(x)`. Sie ist die sogenannte Treppenfunktion, mit deren Hilfe gemäß $x - \text{floor}(x) \in [0,1]$ bzw. $x - \text{floor}(x) - 1 \in [-1,0]$ die gewünschten "Sägezähne" entstehen. Nimmt man an Stelle von x eine andere Funktion, so kann damit schon eine horizontale Stauchung erfolgen. Vertikale Veränderungen erreicht man durch Dämpfungsfunktionen wie z. B. e^{-x} oder $a - x$, falls $0 \leq x \leq a$ ist.
Wir geben einige geeignete "Baumfunktionen" an.

Rechnungen in Maple

4 Baumfunktionen

1. Funktion

```
> f1:=x->tan(x)-1-floor(tan(x)); # eine Baumseite
  f2:=x->-f1(x);                 # und Spiegelung
```

$$f1 := x \to \tan(x) - 1 - \mathrm{floor}(\tan(x))$$
$$f2 := x \to -\mathrm{f1}(x)$$

```
> p1:=plot([tan(x)-1,f1(x),f2(x)],x=0..Pi/2,y=-1..4,thickness=[1,2,2],
            color=[black,green,green]):
  display(p1);
```

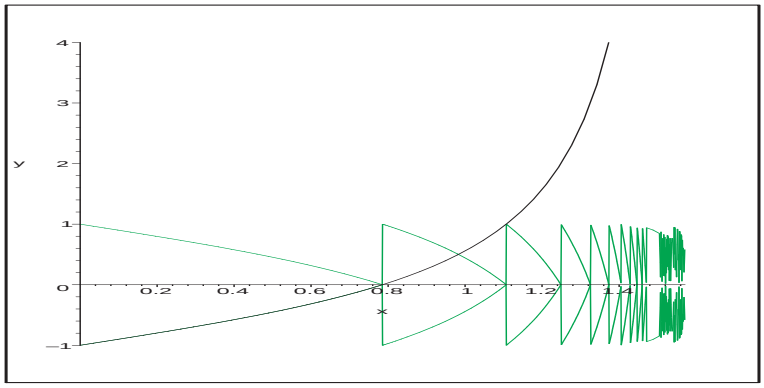


Abb. 2.15 Grundlagen für Baumfunktion

Verjüngung des Baums zur Spitze hin

```
> f3:=x->(Pi/2-x)*(0.9*f1(x)-0.1);
  f4:=x->(Pi/2-x)*(0.9*f2(x)+0.1);
```

$$f3 := x \to \left(\frac{\pi}{2} - x\right)(0.9\,\mathrm{f1}(x) - 0.1)$$
$$f4 := x \to \left(\frac{\pi}{2} - x\right)(0.9\,\mathrm{f2}(x) + 0.1)$$

```
> p2:=plot([f3(x),f4(x)],x=0..Pi/2,y=-Pi/2..Pi/2,thickness=[2,2],
            color=[green,green]):
  plots[display](p2);
```

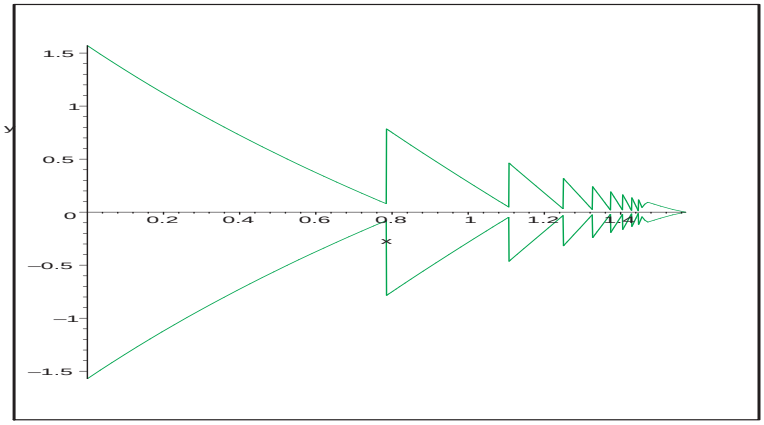


Abb. 2.16 Verjüngung des Baums zur Spitze hin

Aufrichten des Baums

```
> p3:=plot([f3(x),x,x=0..Pi/2],thickness=3,color=green):
  display(p3,axes=none);
```

Gesamtgestaltung des Baums mit Stamm und Schmuck

```
> q1:=plot([f3(x),x,x=0..Pi/2],thickness=3,color=green):
  q2:=plot([f4(x),x,x=0..Pi/2],thickness=3,color=green):
  q3:=plot(0,x=-Pi/2..Pi/2,thickness=3,color=green):
  q4:=plot(0.62+sqrt(0.01-(x-0.75)^2),x=0.65..0.85,thickness=3,color=red):
  q5:=plot(0.62-sqrt(0.01-(x-0.75)^2),x=0.65..0.85,thickness=3,color=red):
  q6:=plot([[-0.1,0],[-0.1,-0.3],[0.1,-0.3],[0.1,0]],
           thickness=4,color=brown):
  q7:=plot([[0.75,0.73],[0.75,0.81]],thickness=3, color=red):

> p4:=plots[display](q1,q2,q3,q4,q5,q6,q7,axes=none,
           title=‘Frohe Weihnachten/Ostern‘):
  display(p4);
```

Abb. 2.17 Aufrichten des Baums sowie
Gesamtgestaltung des Baums mit Stamm und Schmuck

Man probiere es selbst.

2. Funktion

```
> g1:=x->cot(x)-floor(cot(x));
  g2:=x->-g1(x);
```

$$g1 := x \to \cot(x) - \text{floor}(\cot(x))$$
$$g2 := x \to -\text{g1}(x)$$

```
> plot([cot(x),g1(x),g2(x)],x=0..Pi/2-1E-4,y=-1..4,thickness=[1,2,2],
       color=[black,green,green]);
```

3. Funktion

```
> h1:=x->1/x-floor(1/x);
  h2:=x->-h1(x);
```

$$h1 := x \to \frac{1}{x} - \text{floor}\left(\frac{1}{x}\right)$$
$$h2 := x \to -\text{h1}(x)$$

```
> plot([1/x,h1(x),h2(x)],x=0..1,y=-1..4,thickness=[1,2,2],
        color=[black,green,green]);
```

4. Funktion

```
> k1:=x->exp(x)-1-floor(exp(x));
  k2:=x->-k1(x);
```

$$k1 := x \to e^x - 1 - \text{floor}(e^x)$$
$$k2 := x \to -\text{k1}(x)$$

```
> plot([exp(x)-1,k1(x),k2(x)],x=0..2.078,y=-1..4,thickness=[1,2,2],
        color=[black,green,green]);
```

2.2.9 Verhältnis von Fläche zu Umfang

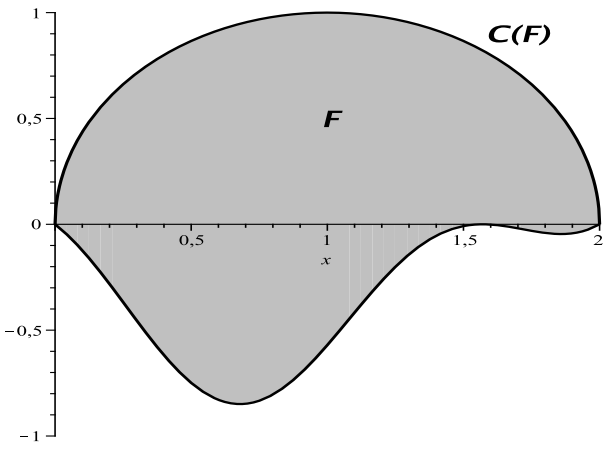
Verzeichnis: `verschiedenes`
Dateien: zeige_mal1.mws

Betrachtet wird eine einfach zusammenhängende ebene Fläche F und ihre geschlossene Kontur (Umfang) $C(F) = (x(t), y(t)),\ t \in [0, T]$, mit der Länge L.
Den Flächeninhalt berechnet man mittels der Formel

$$F \ = \ \frac{1}{2}\int_0^T [x(t)y'(t) - x'(t)y(t)]\,dt \ = \ \frac{1}{2}\int_{C(F)} (xdy - ydx),$$

die Länge der Kontur gemäß

$$L = \int_0^T \sqrt{x'(t)^2 + y'(t)^2}\,dt.$$



Es gilt die Ungleichung

$$F \leq \frac{L^2}{4\pi}.$$

Die Beziehung kann man leicht überprüfen für einfache Flächen.

```
# Rechteck mit Seiten a, b
F=a*b
L=2*(a+b)
F<L^2/(4*Pi)

# Kreis mit Radius r
F=Pi*r^2
L=2*Pi*r;
F=L^2/(4*Pi)

# Ellipse mit Halbachsen a und b, a>b
F=Pi*a*b
e1=sqrt(a^2-b^2)    # lineare Exzentrizitaet
eps1=e1/a           # numerische Exzentrizitaet <1
# Umfang und 2 Naeherungsformeln des Umfangs
L=2*Pi*a*(1-sum(eps1^(2*k)/(2*k-1)*product((2*j-1)/(2*j),j=1..k)^2,
          k=1..infinity))
Ln1=Pi*(3/2*(a+b)-sqrt(a*b))
Ln2=Pi/2*(a+b+sqrt(2*(a^2+b^2)))
F=<L^2/(4*Pi), F=<Ln1^2/(4*Pi), F=<Ln2^2/(4*Pi)
```

Rechnungen in Maple

```
> # Flaeche mit Kontur
  p1:=plot(sqrt(1-(x-1)^2),x=0..2,color=gray,filled=true):
  p2:=plot(sqrt(1-(x-1)^2),x=0..2,color=black,thickness=2):
  p3:=plot(-x*(1+sin(3*x))*(2-x)/2,x=0..2,color=gray,filled=true):
  p4:=plot(-x*(1+sin(3*x))*(2-x)/2,x=0..2,color=black,thickness=2):
  p5:=textplot([[1,0.5,‘ F‘],[1.7,0.9,‘C(F)‘]],font=[HELVETICA,BOLD,16]):
  display([p1,p2,p3,p4,p5],view=[0..2,-1..1]);
  ...
> # Koordinatenfunktionen, 0<=t<=T
  x:=unapply(...,t);
  y:=unapply(...,t);
> xs:=unapply(diff(x(t),t),t);
  ys:=unapply(diff(y(t),t),t);

> L:=int(sqrt(xs(t)^2+ys(t)^2),t=0..T);
  eL:=evalf(L);
> F:=1/2*int(x(t)*ys(t)-xs(t)*y(t),t=0..T);
  eF:=evalf(F);
> test:=evalf(eL^2/(4*Pi)-eF);              # Vorzeichen +
```

2.2.10 Populationsmodell: Gras, Hase, Fuchs, Jäger

Verzeichnis: `gras_hasen_fuechse_jaeger`
Dateien: s29_a09_DglSys_GHFJModell.mws, wiese2.pas, wiese2.exe, schule1.ps

Man bezeichnet ein solches Modell auch als Räuber-Beute-Modell oder als ökologisches System. Das vorliegende Ökosystem soll einfach, endlich und abgeschlossen sei.

Es beruht auf folgenden Annahmen:

1. Es wächst Gras.
2. Hasen fressen Gras, die Zahl der Hasen nimmt zu.
3. Füchse fressen Hasen, die Zahl der Füchse nimmt zu.
4. Füchse werden gejagt, sie werden mit einer Trefferwahrscheinlichkeit erlegt.

A Das diskrete Modell

1. Beschreibung
   Das Modell basiert auf einem quadratischen Spielfeld mit $N \times N$ Feldern. Die Spielsteine sind *Leer, Gras, Hase, Fuchs*. Ihre jeweiligen Anzahlen berechnen wir. Das Spielfeld ist initialisiert. So können z.B. alle Felder leer sein oder eine andere Verteilung von Spielsteinen vorliegen.

   Beispiel: $N = 6$

i \ $j \rightarrow$	1	2	3	4	5	$N=6$
1						
2			*Leer*			
3		*Gras*	$A[i,j]$ *Hase*	*Gras*		
4			*Leer*			
5						
$N=6$						

   In jedem Spielzug wird eines der Felder zufällig bestimmt. In Abhängigkeit vom bisherigen Zustand des Feldes und den 4 Feldnachbarn (nur horizontal und vertikal gesehen) werden das Feld und/oder Feldnachbarn verändert.
   Der Fall, dass ein Hase auf dem Feld sitzt, muss noch etwas genauer beschrieben werden. Bei der Umwandlung setzen wir eine Priorität.
   Falls irgendein Nachbar des Hasens Gras ist, dann sollen alle Nachbarn mit Gras zu Hasen werden. Falls der Hase aber keinen Nachbarn Gras hat, dafür aber einen Fuchs zum Nachbarn, wird er zum Fuchs (vom Fuchs gefressen).

Es sind auch andere Regeln der Umwandlung möglich.
All dies kann man durch eine *Umwandlungstabelle* beschreiben. Wir benutzen hier die folgende Strategie.

Nachbarfeld	Erwürfeltes Feld $A[i,j]$ ist			
	Leer	Gras=Grün	Hase=Gelb	Fuchs=Rot
				jeweils mit der Trefferwahrscheinlichkeit
alle Leer	→Gras			→Leer
Gras	→Gras		Nachbarfeld Gras→Hase	→Leer
Hase	→Gras	→Hase		→Leer
Fuchs	→Gras		→Fuchs	→Leer

2. Wachstumsverhalten
Das Wachstumsverhalten der einzelnen Populationen ist wie folgt charakterisiert.

   - Gras ist abhängig von der Anzahl der Leerfelder.
     Wir nehmen an, dass jederzeit genügend Grassamen im Boden ist.
   - Hasen: abhängig von Gras, Hasen und Füchsen.
   - Füchse: abhängig von Hasen, Füchsen und Abschussquote.
   - Jäger: sie sind indirekt durch die Trefferwahrscheinlichkeit enthalten.

   Folgende Initialisierungen sind Sonderfälle.

   - Ohne Hase und Fuchs: alle Spielfelder werden Gras.
   - Mit Hase, ohne Fuchs: alle Spielfelder werden Hase.
   - Ohne Hase, mit Füchsen: nach hinreichend vielen Schussversuchen und bei positiver Trefferwahrscheinlichkeit werden die Füchse erlegt, und alle Spielfelder werden Gras.

   Bei anderen Anfangsbelegungen zeigt das System i. Allg. ein oszillierendes Verhalten der einzelnen Mengen, wobei die 3 Populationen *Gras, Hase* und *Fuchs* phasenversetzt sind.
   Hasen und Füchse können aussterben. Die Wahrscheinlichkeit ist zwar um so kleiner, je größer N ist, andererseits wächst die Wahrscheinlichkeit mit der Länge des Spiels an. Es ist also sinnvoll, die Anzahlen der 3 Populationen zu messen, aber wegen ihrer starken Oszillationen nur gleitende Durchschnitte über mehrere Spielzüge aufzuzeigen.

3. Simulation
   Betrachten wir einen typischen Simulationsverlauf.
   Das Spielfeld sei initialisiert zur Hälfte mit Gras und einigen Hasen und wenigen Füchsen. Wegen der hohen Verfügbarkeit von Nahrung wächst die Hasenpopulation anfangs stark an. Mit gewisser Verzögerung steigt dann aber auch die Fuchspopulation an (einige werden aber geschossen), und zusammen mit der reduzierten Menge Gras führt dies zu einer Trendumkehr bei der Hasenpopulation. Ist die Hasenpopulation weit genug reduziert, fängt auch die Fuchszahl an zu fallen. Da sich nun auch die Grasmenge wieder erholt hat, kann die Zahl der Hasen erneut ansteigen.
   Das System zeigt also qualitativ ein oszillierendes Verhalten, wobei die 3 Populationen phasenversetzt sind. Jedoch ist das Modell viel zu grob, um quantitative Vorhersagen für ein reales Ökosystem zu finden. Es demonstiert, dass ein System trotz der Zufälligkeit jeder einzelnen Aktion doch in der Gesamtheit zu einem deterministischen Verhalten führen kann.

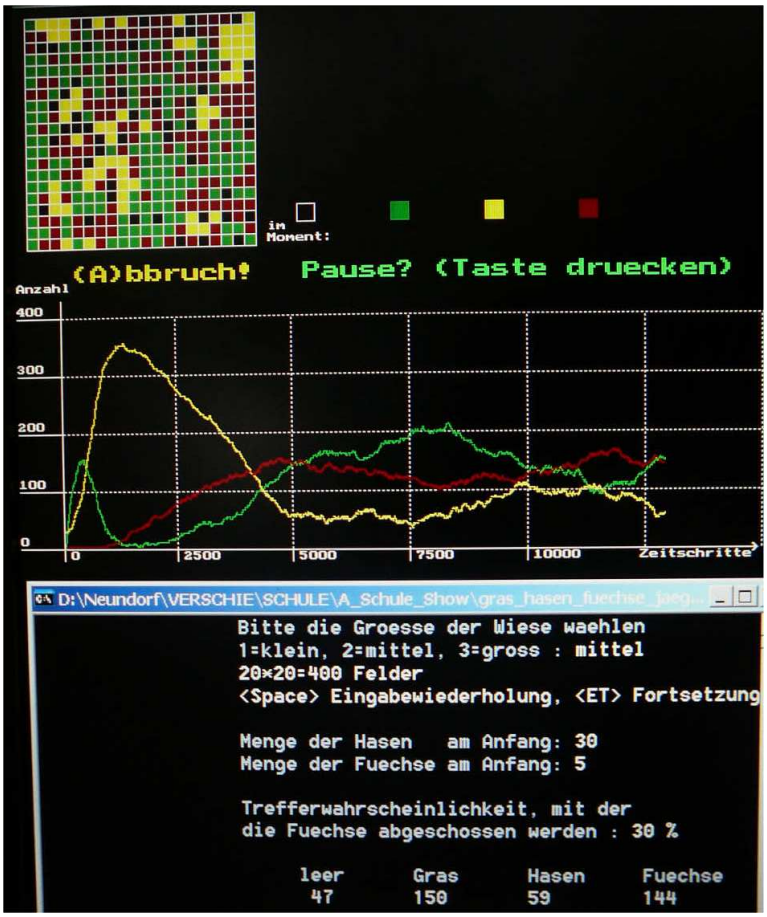


Abb. 2.18 Beispiel für typischen Simulationsverlauf im Ökosystem, Zahl der Individuen geglättet

4. Mathematische Simulation und Programmierung in Pascal

   - Darstellung der Belegung und Entwicklung des Spielfeldes $A[1..N, 1..N]$. N, Umwandlungstabelle und Trefferwahrscheinlichkeiten $T \in [0, 1]$ sind vorgegeben.
   - Darstellung der Gesamtzahlen der Populationen über vorgegebene Anzahl von Schritten. Plot von gleitenden Durchschnitten von jeweils m aufeinanderfolgenden Zeitschritten ($1 \leq m \leq 64$).
   - Wahl des sprachtypischen Zufallszahlengenerators für Spielfeld und Trefferwahrscheinlichkeit.

5. Mögliche Erweiterungen des Modells

   - Einbeziehung weiterer Spezies in das Modell, z.B. Jäger, Heuschrecken.
   - Untersuchung der räumlichen Verteilung der Spezies oder von Wanderungsverhalten, Schädlingsausbreitung.

B Modellierung des Ökosystems mittels Differentialgleichungen

Das typische Räuber-Beute-Modell kann auf der Grundlage von gewöhnlichen Differentialgleichungen (DGL) bzw. von Systemen solcher beschrieben werden.
Dazu nimmt man an, dass
- die räumliche Verteilung der Spezies keine Rolle spielt,
- die Modellierung mit reellen Zahlen aufgrund einer hohen Anzahl von Individuen sinnvoll ist,
- eine Normierung der Anzahlen auf 1 vorgenommen wird.

Bezeichnet man die Menge Gras, die Zahl der Hasen und die Zahl der Füchse mit g, h bzw. f, so kann man folgendes Modell als System von nichtlinearen autonomen DGL 1. Ordnung in Abhängigkeit von der Zeit aufstellen.

$$\begin{aligned} g'(t) &= 1 - h(t)\, g(t), \\ h'(t) &= h(t)(g(t) - f(t)) - c_3 h(t), \\ f'(t) &= f(t)\, h(t) - c_1 f(t) - c_2 \sqrt{f(t)}, \\ & \qquad c_1 + c_2 = c = \text{const} > 0,\ c_i \geq 0. \end{aligned}$$

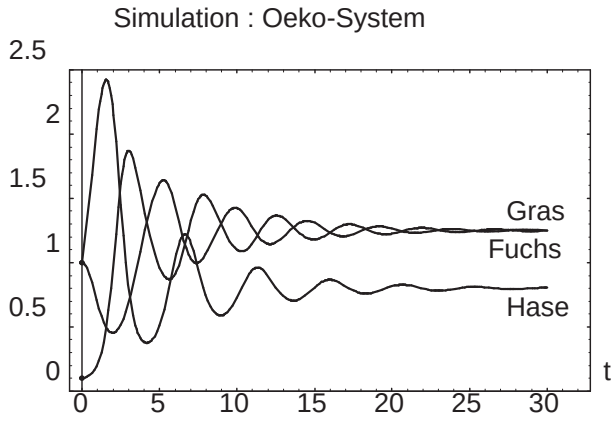
Mögliche Anfangsbedingungen (AB) sind z.B. $g(0) = h(0) = 1,\ f(0) = \frac{1}{10}$.
Die erste Gleichung beschreibt das Wachstum der Grasmenge und das "Gefressenwerden" von Gras mit der Rate, die proportional zum Produkt aus Hasen und Gras ist. In der zweiten Gleichung für Hasen ist die Wachstumsrate ebenfalls mit $h(t)g(t)$ gegeben. Weiter geht die Zahl der Füchse "hasenmindernd" mit $-h(t)f(t)$ sowie zusätzlich die Hasenjagd mit $-c_3 h(t)$ ein. In der Gleichung der Füchse ist ihr Wachstum von den vorhandenen Hasen abhängig, während der negative Term das Jägerverhalten simulieren soll. Der Abschuss erfolgt proportional zur Anzahl der Füchse bzw. Hasen.

1. Einfache Situationen und Lösungen
   (1) $h = 0,\ f = 0,\ g = t$ Gras wächst,
   (2) $h = 0,\ f$ nimmt ab, g nimmt zu,
   (3) $g = 0$ (ohne 1. DGL), h nimmt ab, f nimmt zu und dann ab.

2. Simulationsläufe

   Das System tendiert zu Schwingungen. Diese werden jedoch mit der Zeit herausgedämpft, und es stellt sich ein Gleichgewicht ein.

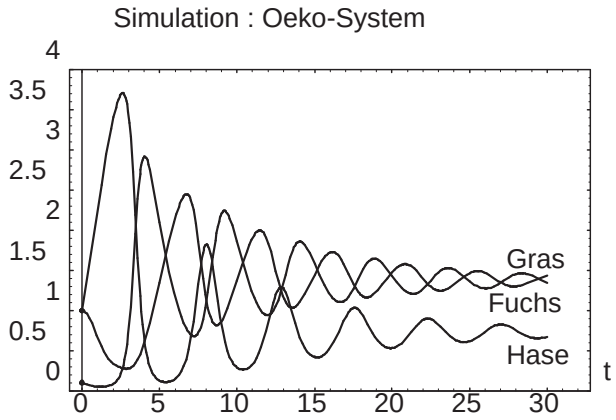
   Beispiel 1: $g(0) = h(0) = 1,\ f(0) = \frac{1}{10},$
   $c_3 = 0,\ c = c_1 = 0.8,\ c_2 = 0$



Man erkennt auch den Trend zu einer Gleichgewichtslage mit den Grenzwerten $g(\infty) = \frac{5}{4}, f(\infty) = \frac{5}{4}, h(\infty) = \frac{4}{5}$. Dies ist zugleich eine stationäre Lösung.

Beispiel 2: $g(0) = h(0) = 1,\ f(0) = \frac{1}{10},$
$c_3 = 0,\ c = 0.8,\ c_1 = 0.3,\ c_2 = 0.5$

Wegen $1 > \sqrt{f(t)} > f(t)$ am Anfang haben wir eine höhere Abschussquote der Füchse als im ersten Beispiel. Die Zahl der Füchse wird so stark dezimiert, so dass es zu einer richtigen Hasenplage kommt. Die Folge dazu ist wieder ein starker Rückgang der Grasmenge. Insgesamt sind die Ausschläge größer.



Man erkennt auch hier den Trend zu einer Gleichgewichtslage mit den Grenzwerten $g(\infty) = \frac{1}{18}(85 - 5\sqrt{145}) = 1.377334838$, $f(\infty) = g(\infty)$, $h(\infty) = \frac{1}{40}(17 + 5\sqrt{145}) = 0.7260398645$.

Dies ist wiederum eine stationäre Lösung.

3. Gleichgewichtslage, stationäre Lösung
   Bei hinreichend großer Zeit stellt sich für die Anzahlen der Populationen ein Gleichgewicht ein. Man berechnet die Gleichgewichtsgrößen $g_\infty = g(t = \infty)$, h_∞ und f_∞ aus den Bedingungen des stationären Verhaltens $g' = h' = f' = 0$. Dabei kann man auch den Typ der Gleichgewichtslage untersuchen.

4. Numerik
   Es gibt zahlreiche Verfahren zur numerischen Lösung des DGL-Systems 1. Ordnung. Man wählt die Parameter c_i und geeignete AB, berechnet die Näherungslösungen und kontrolliert ihren Verlauf. Die Lösungen lassen sich grafisch darstellen.

Rechnungen in Maple

```
> # Loesung des AWP, Loesungstrajektorie, Anfangsbedingung
  t0:=0;
  g0:=1; h0:=1; f0:=1/10;
  c1:='c1': c2:='c2': c3:='c3':
  # Beispiel 1: c1:=4/5;  c2:=0;    c3:=0;
  # Beispiel 2: c1:=3/10; c2:=5/10; c3:=0;
  # weitere Faelle, auch mit c3>0

  AB:=[g0,h0,f0];
  AB1:=[t0,g0,h0,f0];
  ABa:=g(t0)=g0,h(t0)=h0,f(t0)=f0;

> # rechte Seiten des DGL-Systems
  f1:=unapply(1-h*g,g,h,f,c1,c2,c3);
  f2:=unapply(h*(g-f-c3),g,h,f,c1,c2,c3);
  f3:=unapply(f*h-c1*f-c2*sqrt(f),g,h,f,c1,c2,c3);

> # DGl-System
  sys3:=diff(g(t),t)=f1(g(t),h(t),f(t),c1,c2,c3),
        diff(h(t),t)=f2(g(t),h(t),f(t),c1,c2,c3),
        diff(f(t),t)=f3(g(t),h(t),f(t),c1,c2,c3);
  fcns:=g(t),h(t),f(t):
  init:=g(t0)=g0,h(t0)=h0,f(t0)=f0;
```

Beispiel 1

```
> c1:=4/5; c2:=0; c3:=0;
  f1(g,h,f,c1,c2,c3);
  f2(g,h,f,c1,c2,c3);
  f3(g,h,f,c1,c2,c3);
  # stationare Loesung, Fixpunkt, stabiler Strudel
  fp:=solve({f1(g,h,f,c1,c2,c3),f2(g,h,f,c1,c2,c3),f3(g,h,f,c1,c2,c3)},
            [g,h,f]);
> fpu:=vector(3,[]):
  hh:=op(fp);
  fpu[1]:=[rhs(hh[1]),rhs(hh[2]),rhs(hh[3])];
  fpu1:=fpu[1];
```

Phasenportrait zum DGL-System, Szeneauswahl,
keine Richtungsfelder bei Dimension ≥ 3

```
> # Loesungskurven
  p11:=phaseportrait({sys3},[fcns],t=-1..30,[[init]],stepsize=0.01,
                  linecolor=green,scene=[t,g(t)]):
  p12:=textplot([[8,1.8,`g(t) Gras`],[25,1.4,`Gras -> 5/4`]]):
  p13:=pointplot([t0,g0],symbol=solidcircle,symbolsize=20,color=green):
  p21:=phaseportrait({sys3},[fcns],t=-1..30,[[init]],stepsize=0.01,
                  linecolor=yellow,scene=[t,h(t)]):
  p22:=textplot([[2.8,2.6,`h(t) Hasen`],[25,0.7,`Hasen -> 4/5`]]):
  p23:=pointplot([t0,h0],symbol=solidcircle,symbolsize=20,color=yellow):
  p31:=phaseportrait({sys3},[fcns],t=-1..30,[[init]],stepsize=0.01,
                  linecolor=red,scene=[t,f(t)]):
  p32:=textplot([[4.5,0.2,`f(t) Fuechse`],[25,1.1,`Fuechse -> 5/4`]]):
  p33:=pointplot([t0,f0],symbol=solidcircle,symbolsize=20,color=red):

> display(p11,p12,p13,p21,p22,p23,p31,p32,p33,
          view=[-1..30,0..3],labels=[`t`,` `]);


> # Phasenkurven
  p11:=phaseportrait({sys3},[fcns],t=0..30,[[init]],stepsize=0.01,
                  linecolor=green,scene=[g(t),h(t)]):
  p12:=textplot([[0.8,2.6,`(g(t),h(t)) -> (5/4,4/5)`]]):
  p13:=pointplot([g0,h0],symbol=solidcircle,symbolsize=20,color=green):
  p21:=phaseportrait({sys3},[fcns],t=0..30,[[init]],stepsize=0.01,
                  linecolor=yellow,scene=[g(t),f(t)]):
  p22:=textplot([[0.6,0.3,`(g(t),f(t)) -> (5/4,5/4)`]]):
  p23:=pointplot([g0,f0],symbol=solidcircle,symbolsize=20,color=yellow):
  p31:=phaseportrait({sys3},[fcns],t=0..30,[[init]],stepsize=0.01,
                  linecolor=red,scene=[h(t),f(t)]):
  p32:=textplot([[2.4,1,`(h(t),f(t)) -> (4/5,5/4)`]]):
  p33:=pointplot([h0,f0],symbol=solidcircle,symbolsize=20,color=red):

> display(p11,p12,p13,p21,p22,p23,p31,p32,p33,
          view=[0..3,0..3],labels=[``,``]);


> # 3D-Phasenkurve, scene=[g(t),h(t),f(t)]
  # Ausdehnung des Gebiets richtet sich nach "Dimension" der Phasenkurve
  p3:=DEplot3d({sys3},[fcns],t=0..30,[[init]],linecolor=blue,thickness=2,
           stepsize=0.01,labels=[`g`,`h`,`f`],orientation=[60,45]):
  p4:=pointplot3d([g0,h0,f0],symbol=solidcircle,
                   symbolsize=20,color=blue):
  p5:=textplot3d([[1,1,0.2,` [t0,g0,h0,f0]=[0,1,1,1/10]`],
           [1,1,1,` (g(t),h(t),f(t)) mit AB [t0,g0,h0,f0]`]],
           font=[HELVETICA,BOLD,8],color=blue):
  p6:=pointplot3d(fpu1,symbol=solidcircle,symbolsize=20,color=black):

> display(p3,p4,p5,p6);
```

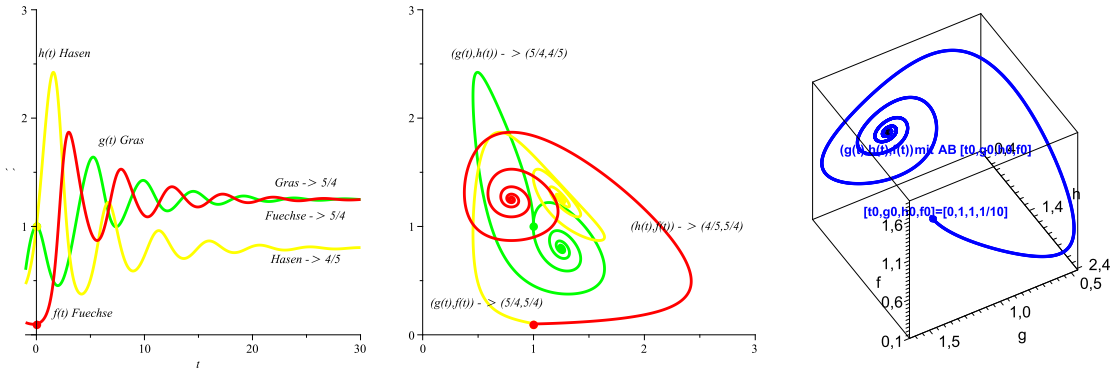


Abb. 2.19 Lösungs- und Phasenkurven im 2D, 3D,
$c_1 = \frac{4}{5},\ c_2 = 0,\ c_3 = 0,$ AB $g(0) = 1,\ h(0) = 1,\ f(0) = \frac{1}{10}$

Beispiel 2
Die Vorgehensweise ist analog zum Beispiel 1

```
> c1:=3/10; c2:=5/10; c3:=0;
  f1(g,h,f,c1,c2,c3);
  f2(g,h,f,c1,c2,c3);
  f3(g,h,f,c1,c2,c3);
  # stationare Loesung, Fixpunkt, stabiler Strudel
  fp:=solve({f1(g,h,f,c1,c2,c3),f2(g,h,f,c1,c2,c3),f3(g,h,f,c1,c2,c3)},
             [g,h,f]);
> fpu:=vector(3,[]):
  hh:=op(fp);
  fpu[1]:=[rhs(hh[1]),rhs(hh[2]),rhs(hh[3])];
  fpu2:=fpu[1];

> p11:=phaseportrait({sys3},[fcns],t=-1..30,[[init]],stepsize=0.01,
                     linecolor=green,scene=[t,g(t)]):
  p12:=textplot([[8,2.6,'g(t) Gras'],[25,1.7,'Gras -> 1.377']]):
  p13:=pointplot([t0,g0],symbol=solidcircle,symbolsize=20,color=green):
  ...
```

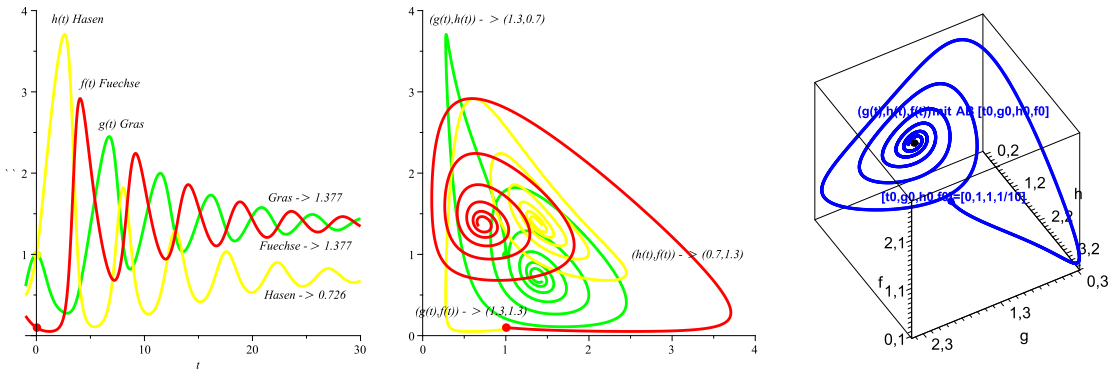


Abb. 2.20 Lösungs- und Phasenkurven im 2D, 3D,
$c_1 = \frac{3}{10},\ c_2 = \frac{5}{10},\ c_3 = 0,$ AB $g(0) = 1,\ h(0) = 1,\ f(0) = \frac{1}{10}$

Beispiel 3 $g(0) = h(0) = 1,\ f(0) = 0.1,$
$c_3 = 0,\ c = 0.3,\ c_1 = 0.1,\ c_2 = 0.2$

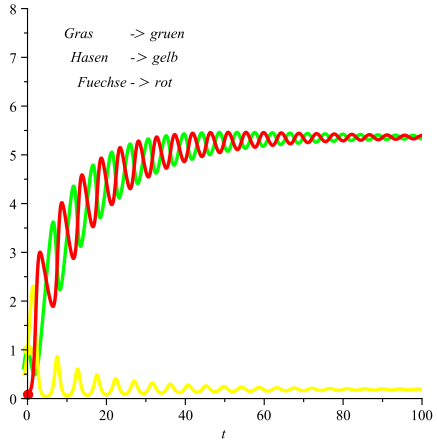


Abb. 2.21
Lösungskurven im 2D
$c_1 = \frac{1}{10},\ c_2 = \frac{1}{5},\ c_3 = 0,$
AB $g(0) = 1,\ h(0) = 1,\ f(0) = \frac{1}{10},$
Gleichgewichtslage
$g_\infty = 12 - 2\sqrt{11} = 5.366750420,$
$f_\infty = g_\infty,$
$h_\infty = \frac{1}{50}(6 + \sqrt{11}) = 0.1863324958$

Beispiel 4 $g(0) = h(0) = 1,\ f(0) = 0.1,$
$c_3 = 0.01,\ c = 0.25,\ c_1 = 0.05,\ c_2 = 0.2$

3 Gleichgewichtslagen (die 1. ist stabil, die beiden anderen instabil)
$g_\infty = 8.3999951351197697, h_\infty = 0.1190476879943743,\ f_\infty = 8.3899951351197697,$
$g_\infty = \frac{1}{100},\ h_\infty = 100,\ f_\infty = 0,$
$g_\infty = 0.0100040072142311, h_\infty = 99.9599439090228147, f_\infty = 0.0000040072142311.$

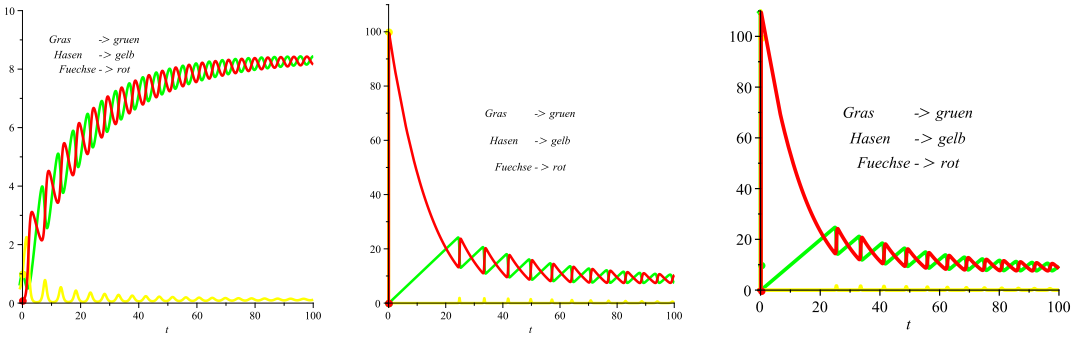


Abb. 2.22 Lösungskurven im 2D, $c_1 = \frac{1}{20},\ c_2 = \frac{1}{5},\ c_3 = \frac{1}{100},$

AB v.l.n.r			
	$g(0) = 1,$	$h(0) = 1,$	$f(0) = 0.1,$
	$g(0) = 0.1,$	$h(0) = 100,$	$f(0) = 0.01,$
	$g(0) = 10,$	$h(0) = 100,$	$f(0) = 0.00001$

Zahlreiche AB erweisen sich als ungeeignet, wie in der letzten Grafik der nächsten Abbildung zu sehen ist. Es kann dann passieren, dass die Rechnung vorzeitig abbricht. Man erhält die Fehlermeldung

```
Warning, plot may be incomplete, the following errors(s) were issued:
cannot evaluate the solution further right of ..., probably a singularity
```

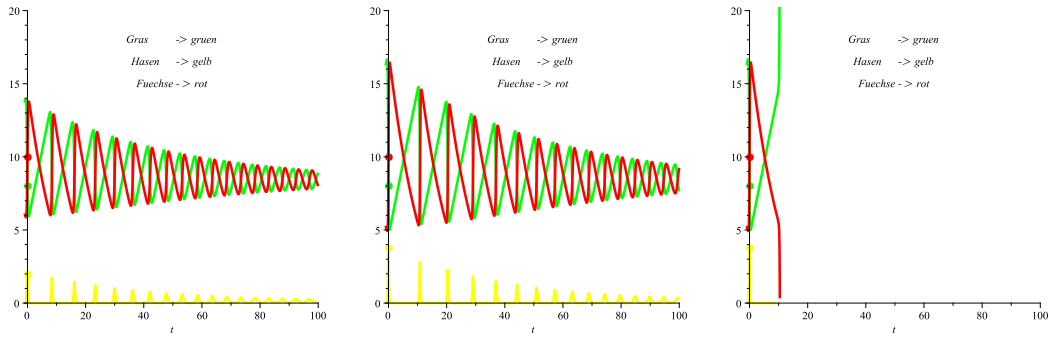


Abb. 2.23 Lösungskurven im 2D, $c_1 = \frac{1}{20}$, $c_2 = \frac{1}{5}$, $c_3 = \frac{1}{100}$,

AB v.l.n.r. $g(0) = 8,\quad h(0) = 2,\quad f(0) = 10,$
$g(0) = 8,\quad h(0) = 3.77,\quad f(0) = 10,$
$g(0) = 8,\quad h(0) = 3.78,\quad f(0) = 10$

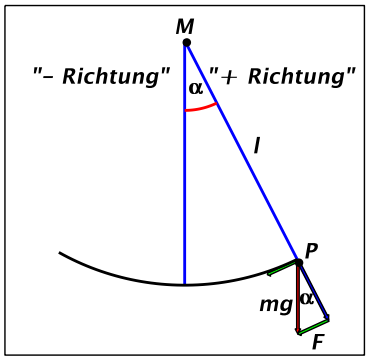
2.2.11 Fadenpendel

Verzeichnis: `schule, kurven_2d`
Dateien: schule3.ps, einfach_pendel1.mws

Wie verläuft die Bewegung eines Pendelkörpers unter den Vereinfachungen:
keine Berücksichtigung des Luftwiderstandes, Masse des Fadens wird vernachlässigt, Pendelkörper ist Massepunkt (mathematisches Pendel).

Bekanntlich gilt nach **I. Newton** $F = m\,a$.
Für kleine Auslenkungen $\alpha(t)$ ergibt sich aus dem Newtonschen Grundgesetz folgende Gleichung für das mathematische Pendel (Kräfteparallelogramm)

$$m\ g\ \sin\alpha(t) = -m\ l\ \alpha''(t),$$



l	Fadenlänge,
$l\,\alpha(t)$	Weg, den das Pendel zurücklegt
	= Länge des Kreisbogens,
$a = l\,\alpha''(t)$	Beschleunigung des Pendelkörpers,
g	Erdbeschleunigung,
$m\,g$	Gewichtskraft,
F und α	haben entgegengesetzte Richtungen.

Für kleine Winkel ist $\sin(\alpha) \approx \alpha$, so dass eine einfache lineare Differentialgleichung (DGL) entsteht.

$$\alpha''(t) + \frac{g}{l}\ \alpha(t) = 0.$$

1. Allgemeine Lösung der DGL
   Hinweis: Finden von Basislösungen mittels Ansatz $e^{\lambda t}$.
   Die beiden unabhängigen Lösungen sind $\sin\left(\sqrt{\frac{g}{l}}t\right)$, $\cos\left(\sqrt{\frac{g}{l}}t\right)$.

Die Lösungsmenge ist ein Vektorraum gebildet mittels der Linearkombination

$$\alpha(t) = c_1 \sin\left(\sqrt{\frac{g}{l}}\ t\right) + c_2 \cos\left(\sqrt{\frac{g}{l}}\ t\right), \quad c_{1,2} \in \mathbb{R}.$$

2. Berücksichtigung von Anfangsbedingungen (AB)
   Es sollen 2 Varianten unterschieden werden.

   - Pendel wird nach Auslenkung um einen Winkel α_0 losgelassen.
     Wenn zur Zeit $t = 0$ die AB $\alpha(0) = \alpha_0$ und $\alpha'(0) = \alpha'_0 = 0$ bekannt sind, ergeben sich die Koeffizienten zu $c_1 = 0,\ c_2 = \alpha_0$. Daraus folgt

     $$\alpha(t) = \alpha_0\ \cos\left(\sqrt{\frac{g}{l}}\ t\right).$$

   - Pendel wird aus der Lage $\alpha(0) = 0$ angestoßen mit der Geschwindigkeit $v_0 = l\alpha'(0)$. Wenn zur Zeit $t = 0$ die AB $\alpha(0) = 0$ und $\alpha'(0)$ bekannt sind, ergeben sich die Koeffizienten zu $c_2 = 0,\ c_1 = \sqrt{l/g}\ \alpha'(0)$. Daraus folgt

     $$\alpha(t) = \sqrt{\frac{l}{g}}\ \alpha'(0)\ \sin\left(\sqrt{\frac{g}{l}}\ t\right).$$

3. Lösung der DGL 2. Ordnung mittels Polygonzugverfahren (PZV) als Näherungsverfahren. Das Anfangswertproblem

   $$\alpha''(t) = -\frac{g}{l}\ \alpha(t), \quad \alpha(0),\ \alpha'(0) \text{ gegeben},\ t \in [0, T]$$

   kann man mittels $\alpha'(t) = \beta(t)$ auf ein System von zwei DGL 1. Ordnung transformieren.

   $$\begin{aligned} \alpha'(t) &= \beta(t), && \alpha(0) \text{ gegeben}, \\ \beta'(t) &= \alpha''(t) = -\frac{g}{l}\ \alpha(t), && \beta(0) = \alpha'(0) \text{ gegeben}. \end{aligned}$$

   Nunmehr wendet man auf das allgemein notierte DGL-System

   $$\begin{aligned} x' &= f(t, x, y), \quad x(0) \text{ gegeben}, \\ y' &= g(t, x, y), \quad y(0) \text{ gegeben} \end{aligned}$$

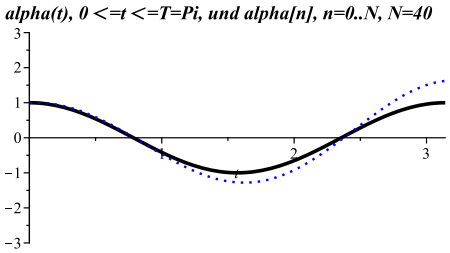
   das PZV an.

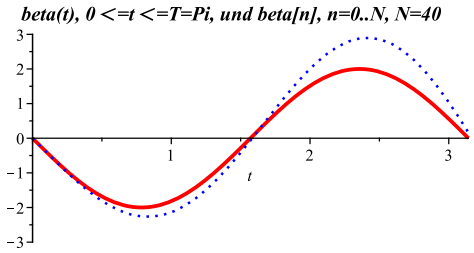
   $$\begin{aligned} x_{n+1} &= x_n + hf(t_n, x_n, y_n),\ x_0 = x(0), \quad n = 0, 1, ...,\ t_n = nh,\ h > 0, \\ y_{n+1} &= y_n + hg(t_n, x_n, y_n), \quad y_0 = y(0). \end{aligned}$$

   Man löse das Anfangswertproblem im Intervall $[0, T]$ mit verschiedenen Schrittweiten und ausgewählten AB und stelle die Näherungslösung tabellarisch sowie grafisch dar.
   Man vergleiche diese mit der exakten Lösung.

4. Zwei Varianten der Darstellung (gepunktete Kurven sind Näherungen)
   Parameter

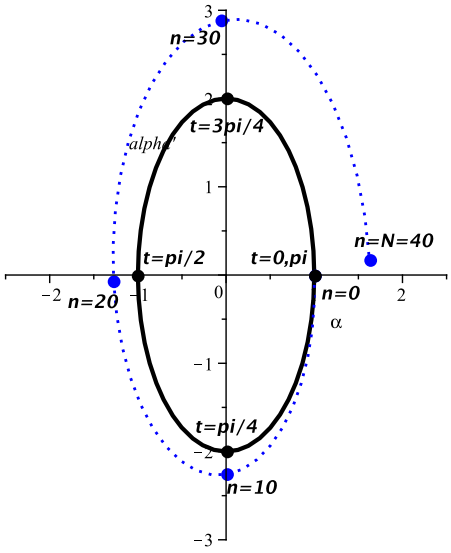
$$\sqrt{\frac{g}{l}} = 2,\ \alpha_0 = 1,\ \alpha_0' = 0,\ t \in [0,T] = [0,\pi],\ N = 40,\ h = T/N.$$





- Vergleichende Grafik für Näherung $\{\alpha_0, \alpha_1, ..., \alpha_N\}$ (Polygonzug) und exakte Lösung $\alpha(t)$.
  Illustration am Beispiel $\alpha(t) = \alpha_0 \cos\left(\sqrt{\frac{g}{l}}\, t\right)$.
- Vergleichende Grafik für Phasenkurven
  - der Näherung (Polygonzug) $\{(\alpha_0, \alpha_0'), (\alpha_1, \alpha_1'), ..., (\alpha_N, \alpha_N')\}$, $\alpha' = \beta$, und
  - der exakten Lösung $(\alpha(t), \alpha'(t))$.
  Illustration am Beispiel

$$\begin{aligned} \alpha(t) &= \alpha_0 \cos\left(\sqrt{\frac{g}{l}}\, t\right), \\ \alpha'(t) &= -\alpha_0 \sqrt{\frac{g}{l}} \sin\left(\sqrt{\frac{g}{l}}\, t\right). \end{aligned}$$



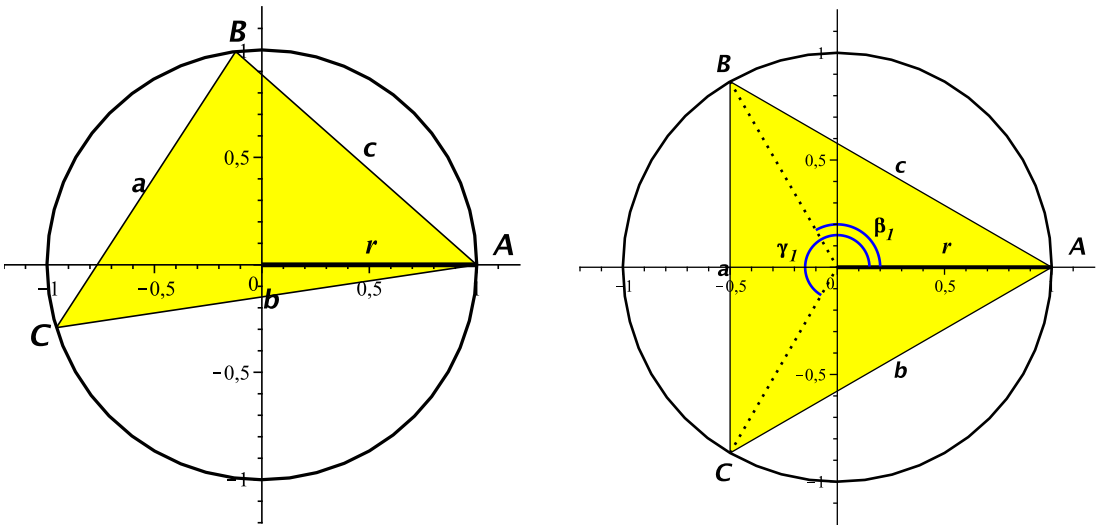
2.3 Abstände, Entfernungen und Flächen

2.3.1 Dreieck mit größter/kleinster Fläche

Verzeichnis: `minimal`
Dateien: minflaeche.mws, zeige_mal1.mws

A Dreieck mit größter Fläche

Gegeben sei ein Kreis. Unter allen Dreiecken mit Eckpunkten auf dem Kreisumfang hat das gleichseitige Dreieck die größte Fläche.
Darstellung der Situation



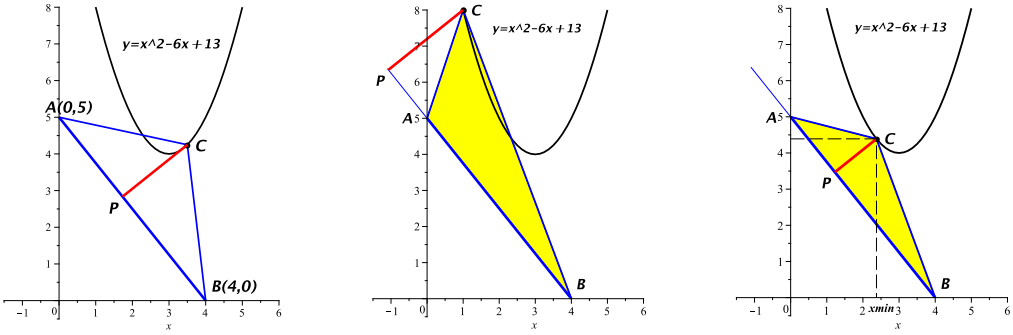
Bestätigung der Aussage durch Lösung eines Extremalproblems

```
> # Eckpunkte des Dreiecks in Abhaengigkeit von den Winkeln beta1,gamma1
  A:=[r,0];
  Ak:=[A[1]*cos(A[2]),A[1]*sin(A[2])];
  B:=[r,beta1/180*Pi];          # beta1=Winkel(A,0,B)
  Bk:=unapply([B[1]*cos(B[2]),B[1]*sin(B[2])],beta1);
  C:=[r,gamma1/180*Pi];         # gamma1=Winkel(A,0,C)
  Ck:=unapply([C[1]*cos(C[2]),C[1]*sin(C[2])],gamma1);

  # Seiten des Dreiecks
  a:=sqrt((Bk(beta1)[1]-Ck(gamma1)[1])^2+(Bk(beta1)[2]-Ck(gamma1)[2])^2);
  b:=sqrt((Ak[1]-Ck(gamma1)[1])^2+(Ak[2]-Ck(gamma1)[2])^2);
  c:=sqrt((Bk(beta1)[1]-Ak[1])^2+(Bk(beta1)[2]-Ak[2])^2);

  # Flaeche des Dreiecks
  s:=0.5*(a+b+c);
  F:=unapply(sqrt(s*(s-a)*(s-b)*(s-c)),beta1,gamma1):

  # --> Maximum  F(beta1,gamma1)=r*3*sqrt(3)/4=r*1.299
```

B Dreieck mit kleinster Fläche

Gegeben seien die Eckpunkte $A = (0, 5)$ und $B = (4, 0)$ eines Dreiecks ABC.
Der Eckpunkt C bewegt sich auf der Kurve (quadratische Parabel) $y = x^2 - 6x + 13$.
Finde unter allen Dreiecken ABC das Dreieck mit der kleinsten Fläche F.

Darstellung der Situation



Das flächenkleinste Dreieck (rechtes Bild) hat die Parameter

$$x_{min} = \frac{19}{8} = 2.375,\ h_{min} = 1.473890,\ F_{min} = 4.718750.$$

Rechnungen mit Maple

```
> x:='x':
  y:=x->x^2-6*x+13;
  A:=[0,5];
  B:=[4,0];
  C:=x->[x,y(x)];
  x0:=3.5;
  C(x0);
  g:=x->4/5*(x-x0)+C(x0)[2];
  c:=x->-5/4*(x-4);
  p:=solve(c(x)=g(x));
  P:=[p,c(p)];
> # Hoehe h des Dreiecks (linkes Bild)
  h:=evalf(sqrt((P[1]-C(x0)[1])^2+(P[2]-C(x0)[2])^2));    # 2.264519547
  # Seite c des Dreiecks
  cs:=evalf(sqrt((A[1]-B[1])^2+(A[2]-B[2])^2));           # 6.403124237
  # Flaeche des Dreiecks
  F:=cs*h/2;                                              # 7.250000000
> # Vorbereitung der Animation
  p1:=plot(y(x),x=0..6,thickness=2,color=black,scaling=constrained):
  p2:=plot([A,B],thickness=3,color=blue):
  p21:=plot(c(x),x=-1.1..0,color=blue):
  p51:=textplot([[-0.6,5,' A'],[4.2,0.4,' B']],font=[HELVETICA,BOLD,13]):
  p71:=textplot([3,7.5,' y=x^2-6x+13'],font=[HELVETICA,BOLD,10]):
  pp1:=display([p1,p2,p21,p51,p71],view=[-1.5..6,-0.5..8]):
  Fmin:=1e10: xmin:=10:
  pl:=vector(41,[]):
  i:=1:
  for x0 from 1 by 0.1 to 5 do
```

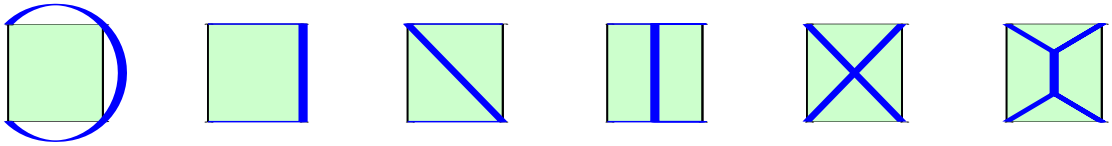
```
    C(x0);
    g:=x->4/5*(x-x0)+C(x0)[2];
    p:=solve(c(x)=g(x));
    P:=[p,c(p)];
    p52:=textplot([[C(x0)[1]+0.3,C(x0)[2],‘ C‘],[p-0.3,c(p)-0.3,‘ P‘]],
                  font=[HELVETICA,BOLD,13]):
    p3:=plot([A,C(x0),B],thickness=2,color=blue):
    p4:=polygon([A,C(x0),B],color=yellow):
    p6:=pointplot(C(x0),symbol=solidcircle,symbolsize=16):
    p7:=plot(g(x),x=p..x0,thickness=3,color=red,linestyle=1):
    pp2:=display([p3,p4,p52,p6,p7],view=[-1.5..6,-0.5..8]);
    pl[i]:=display(pp1,pp2); i:=i+1:
    h1:=evalf(sqrt((P[1]-C(x0)[1])^2+(P[2]-C(x0)[2])^2));
    # Flaeche des Dreiecks
    F1:=cs*h1/2;
    if F1<Fmin then
     Fmin:=F1; xmin:=x0;
    end if;
   end do:

> # Animation (mittleres Bild)
  pp:=[seq(pl[i],i=1..41)]:
  plots[display](pp,insequence=true,scaling=constrained);
```

2.3.2 Minimaler Verbindungsweg

Verzeichnis: `minimal`
Dateien: minweg.mws, zeige_mal2.mws

Die Aufgabenstellung liegt in folgender vereinfachter Version vor.
Bestimmung der minimalen Länge eines Verbindungsweges von 4 Orten in einer Ebene, die Ecken eines Quadrats mit der Seitenlänge a sind. Der mathematische Betrachtung des Sachverhalts beinhaltet somit Aspekte der Optimierung und Lösung von Gleichungen (Nullstellenprobleme).
Die Seite des Quadrats sei $a = 1$ (Skalierung).

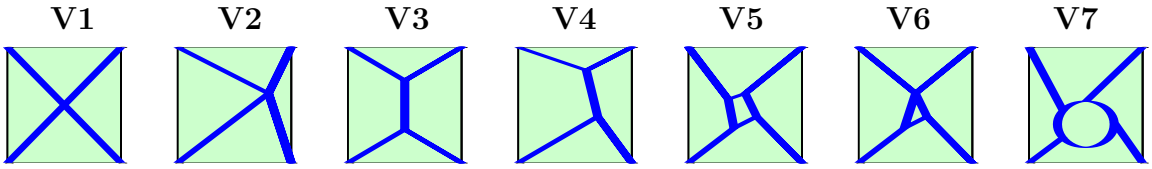
Mögliche einfache Verbindungswege, wie man von jedem Ort aus jeden anderen erreicht, lassen sich sofort angeben. Dabei erkennt man auch günstige und weniger günstige Varianten.



Die zugehörigen 6 Weglängen sind damit (Mantisse mit 3 Nachkommastellen)

$$l_1 = \tfrac{3\sqrt{2}}{4}\pi = 3.332, \quad l_2 = 3, \quad l_3 = 2 + \sqrt{2} = 3.414,$$
$$l_4 = 3, \quad l_5 = 2\sqrt{2} = 2.828, \quad l_6 = 1 + \sqrt{3} = 2.732.$$

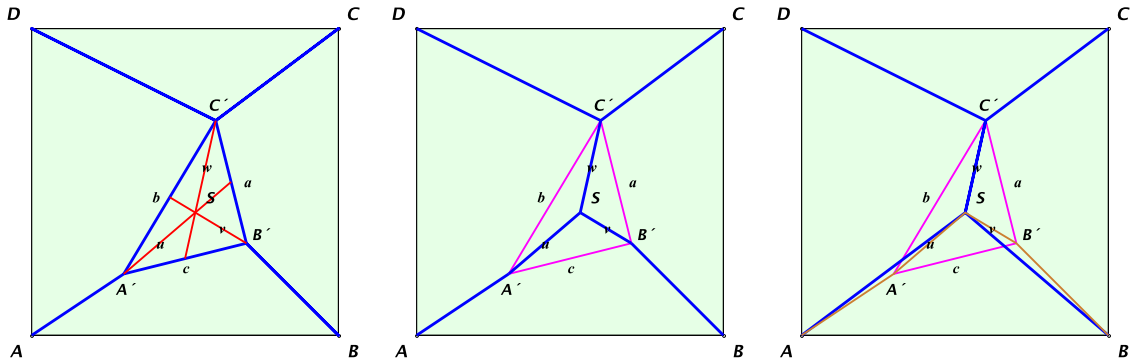
Das letzte Bild ("eingebeultes" Doppel-T) liefert hier den kleinsten Wert und es ist grafisch eine Stellung zwischen den Bildern 4 (Doppel-T) und 5 (Kreuzung).
Zeichnen wir die Verbindungswege etwas komplizierter in 7 Varianten.
Dabei sind noch einmal die Kreuzung V1 und das "eingebeulte" Doppel-T V3.



Wir zeigen, dass die Varianten 5,6,7 als Zugang für die optimale Lösung nicht in Frage kommen.

V7
Die 4 inneren Punkte (Knoten) bilden ein Viereck, dessen Umfang schon kleiner wäre als der Umfang der Ellipse.

V6
Variante 6 ist eigentlich ein Sonderfall von Variante 5, wo 2 innere Knoten zusammenfallen.
Wir transformieren diese Situation in zwei Schritten auf die Variante 4.



Sei S der Schwerpunkt des Dreiecks $A'B'C'$ (Schnittpunkt der Seitenhalbierenden, Teilungsverhältnis der Strecken 2:1, z.B. $u : u' = 2 : 1$).
Das Ersetzen der "alten Teillänge" $a + b + c$ durch $A'S + B'S + C'S = u + v + w$ verkürzt den Weg, denn es gelten

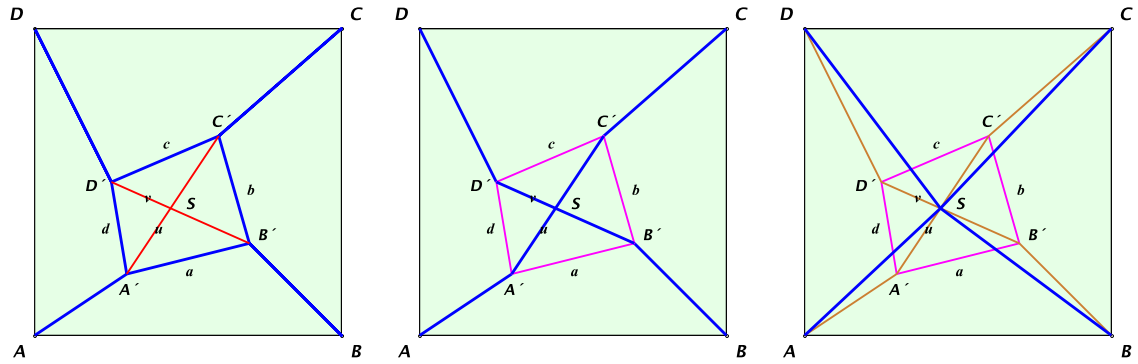
```
u+u' <= c+a/2,   u+u' <= b+a/2,   u' = u/2
v+v' <= a+b/2,   v+v' <= c+b/2,   v' = v/2
w+w' <= b+c/2,   w+w' <= a+c/2,   w' = w/2
-----------------------------------------
2*3/2(u+v+w) <= 3(a+b+c)
       u+v+w <= a+b+c
```

Weiteres Ersetzen von $AA'S$ durch die Strecke AS sowie $BB'S$ durch BS verkürzt noch einmal den Weg. Damit ist man bei Variante 4.

V5

Variante 5 ist eigentlich der allgemeine Fall von Variante 2, wo der innere Knoten in 4 Nachbarknoten zerfällt.
Wir transformieren diese Situation in zwei Schritten auf die Variante 2.

```
> ma:=[0.3,0.2]:
  mb:=[0.7,0.3]:
  mc:=[0.6,0.65]:
  md:=[0.25,0.5]:
> s1:=(mc[2]-ma[2])/(mc[1]-ma[1]):
  s2:=(md[2]-mb[2])/(md[1]-mb[1]):
  xx:= (md[2]-ma[2]+ma[1]*s1-md[1]*s2)/(s1-s2):
  S:=[xx,ma[2]+(xx-ma[1])*s1]:
> pl5:=plot([l1[1],ma,mb,l1[2],mb,mc,l1[3],mc,md,l1[4],md,ma],
            thickness=3,color=blue):
  ph1:=textplot([[ma[1]+0.01,ma[2]-0.05,‘A‘],[mb[1]+0.06,mb[2]+0.03,‘B‘],
                 [mc[1]+0.01,mc[2]+0.05,‘C‘],[md[1]-0.05,md[2]-0.01,‘D‘],
                 [0.5,0.43,‘ S‘]],font=[HELVETICA,BOLD,13]):
  ph2:=plot({[ma,mc],[mb,md]},thickness=2,color=red):
  ph3:=textplot( font=[TIMES,BOLD,13],
           [[op((ma+mb)/2+[0,-0.04]),‘ a‘],[op((mb+mc)/2+[0.05,0]),‘ b‘],
            [op((mc+md)/2+[0,0.05]),‘ c‘],[op((md+ma)/2-[0.05,0]),‘ d‘]]):
  ph4:=textplot([[op((2*ma+mc)/3),‘ u‘],[op((3*md+mb)/4),‘ v‘]],
                font=[TIMES,BOLD,13]):
> display(p1,p2,p3,pl5,ph1,ph2,ph3,ph4);
```



Sei S der Schnittpunkt der Diagonalen des Vierecks $A'B'C'D'$.
Das Ersetzen der “alten Teillänge“ $a+b+c+d$ durch $A'C'+B'D'=u+v$ verkürzt den Weg, denn es gelten

```
u <= a+b,  u <= c+d,
v <= a+d,  v <= b+c
-------------------
2(u+v)<= 2(a+b+c+d)
  u+v <= a+b+c+d
```

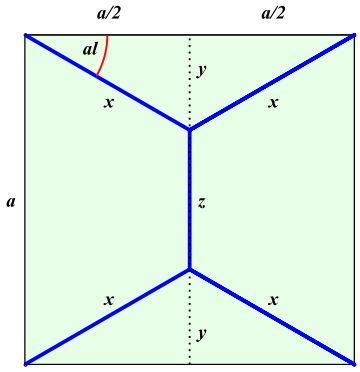
Weiteres Ersetzen von $AA'S$ durch die Strecke AS, $BB'S$ durch BS, $CC'S$ durch CS sowie $DD'S$ durch DS verkürzt noch einmal den Weg.

Untersuchung zur Variante 3

Man beachte zunächst die Symmetrie des Verbindungswegs. Die Funktionen für die Weglänge und daraus für die Minimierung sind

$$\begin{aligned} weg(\alpha, a) &= 4x + z \\ &= 2a/\cos(\alpha) + a(1 - \tan(\alpha)) \\ &= a[1 + (1 - \sin(\alpha)/2)\, 2/\cos(\alpha)] \\ &= a[1 + (2 - \sin(\alpha))/\cos(\alpha)] \end{aligned}$$

$$\begin{aligned} g(\alpha) &= weg(\alpha, a)/a - 1 \\ &= (2 - \sin(\alpha))/\cos(\alpha) \end{aligned}$$



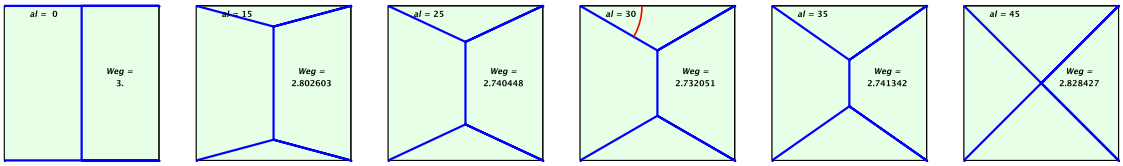
Die Bestimmung des Extremums mittels der nowendigen Bedingung $g'(\alpha) = 0$ liefert den Winkelwert $\alpha = \frac{\pi}{6}$.

Animation zum minimalen Weg

Winkelwerte $\alpha = 0, 1, 2, ..., 45^o$

```
> a:=1:

> pa:=vector(46,[]):

> ph17:=plot([0.4*cos(t),1+0.4*sin(t),t=11*Pi/6..2*Pi],
             color=red,thickness=3):

> for al from 0 to 45 do
   alb:=al*Pi/180;
   xa:=a/2/cos(alb):
   ya:=xa*sin(alb):
   pl3:=plot([l1[1],[1/2,ya],l1[2],[1/2,ya],
              [1/2,1-ya],l1[3],[1/2,1-ya],l1[4]],
              thickness=4,color=blue):
   w:=evalf(weg(alb,a),7);
   ph15:=textplot([[0.75,0.58,`Weg =`],[0.75,0.5,`w`]],
                  font=[HELVETICA,BOLD,12]);
   ph16:=textplot([[0.22,0.95,`al =`],[0.32,0.95,`al`]],
                  font=[HELVETICA,BOLD,12]);
   if al=30 then pa[al+1]:=display(p1,p2,pl3,ph15,ph16,ph17);
            else pa[al+1]:=display(p1,p2,pl3,ph15,ph16);
   end if;
  end do:

> ppa:=[seq(pa[i],i=1..46)]:
  display(ppa,insequence=true,scaling=constrained);
```

Ausgewählte Frames der Animation



Weitere Aufgaben

1. Man überlege, wie man nachweisen kann, dass die Situation V2 immer auf Lösung V1 sowie V4 auf die Minimallösung V3 führen.

2. Ein zweiter Aspekt der Optimierung ist folgender.
   Ziel ist es, die Summe aller "Strecken" zwischen zwei Orten möglichst klein zu machen. Wenn $l(i,j)$ die Weglänge zwischen den Orten i und j ist, dann folgt die Wegsumme

   $$s(a) = l(A,B) + l(A,C) + l(A,D) + l(B,C) + l(B,D) + l(C,D) \quad \to \quad \min\,.$$

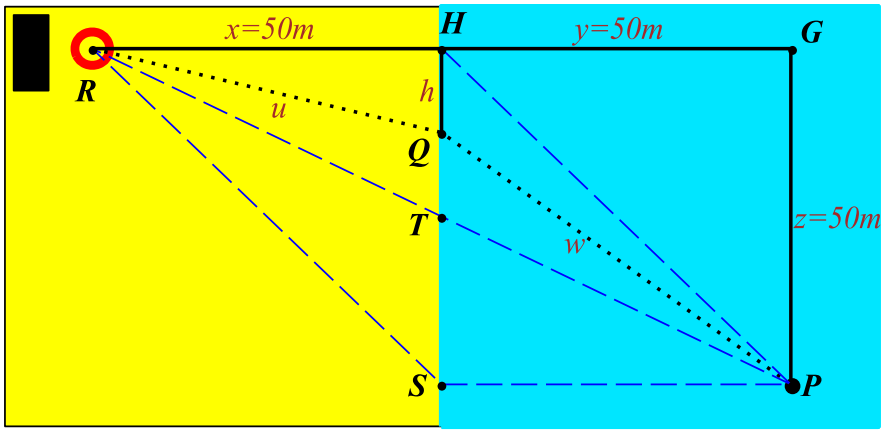
   Man untersuche alle Varianten in Bezug auf diese Bedingung.
   Z.B. $a = 1$,
   V1: $s_1 = \sqrt{2}(3+2+1) = 6\sqrt{2} = 8.484$ mit nur 1 Kreuzung,
   V3: $s_3 = (2x + 2x + z + 2x + z) + (2x + z + 2x + z) + 2x = 4(3x + z)$
   $= 4(1 + 2/3\sqrt{3}) = 8.620.$

3. Praktischer Aspekt unter Berücksichtigung von Ökonomie, Verkehrssicherheit, Langzeitplanung.
   Nach Punkt 2 hat der minimale Weg (V3) $weg(\pi/6, a)$ nicht die minimale Wegsumme $s(a)$. Man denke sich nun folgende Situation.
   Der Auftraggeber AG (Landkreis) hat das Geld für den Bau der kürzesten Weglänge (V3) mit 2 Kreuzungen. Die in der Region ansässige und bauausführende Firma bevorzugt jedoch die Variante 1 wegen der kleineren Wegsumme $s(a)$ und nur einer Kreuzung. Der AG gestattet der Firma den Bau von Variante 1, wenn die entstehenden Mehrkosten von der Firma selber getragen werden.
   Die Firma sagt sich, ich habe in der Region ständig viele Baustellen und meine Arbeiter, von denen sehr viele in den 4 Orten A, B, C, D wohnen und mit Firmenautos zur Arbeit in die Nachbarorte fahren. Über lange Sicht kann ich doch Kraftstoff einsparen und dann meine Mehrkosten kompensieren. Ist die Idee gut?
   Erstelle ein geeignetes Modell und teste es.

2.3.3 Rettungsschwimmerproblem

Verzeichnis: `rettungsschwimmer`
Dateien: rettungsschwimmer.mws, zeige_mal5.mws
Modell: Rettungsring

Stellen wir uns folgende Situation an einem Badestrand vor.
Ein Rettungsschwimmer R steht vor seinem Turm und sieht im Wasser eine Person P in Not. Auf dem direkten Weg zum Wasser sind es $50\,m$. Von dort aus befindet sich die Person nochmals $50\,m$ geradeaus weiter und dann $50\,m$ zur Seite. R weiss, dass er am Ufer $7\,m/s$ schnell ist und im Wasser nur $2\,m/s$. Um schnellstmöglich zu P zu gelangen, rennt er deshalb zuerst am Strand auf geradem Weg zu einem Punkt Q, von wo aus er direkt zur P schwimmt.
Er legt dabei auf dem Strand u Meter und im Wasser w Meter zurück.



Fragen
Wie lange dauert die Rettungsaktion? Zeit $= t$.
Welche Strecke wird zurückgelegt? Weg $= u + w$.
Vergleichen wir einige Varianten.

```
> # Problemgroessen
  x:=50: y:=50: z:=50:   # Entfernungen
  vu:=7: vw:=2:          # Geschwindigkeiten
```

(1) R rennt zuerst $50\,m$ direkt zum Wasser - also zum Punkt H - und schwimmt dann die ca. $71\,m$ lange Strecke zu P.

```
> t1:=x/vu+sqrt(y^2+z^2)/vw;
  evalf(t1);
  w1:=x+sqrt(y^2+z^2);
  evalf(w1);
```

$$t1 := \frac{50}{7} + 25\sqrt{2}$$
$$42.49819619$$
$$w1 := 50 + 50\sqrt{2}$$
$$120.7106781$$

(2) R nimmt die Luftlinie zu P, damit den kürzesten Weg.
Er rennt zuerst schräg zum Wasser - also zum Punkt T - um dann im Wasser zu P zu schwimmen.

```
> s:=x*z/(x+y):           # s=TH
  t2:=sqrt(x^2+s^2)/vu+sqrt(y^2+(z-s)^2)/vw;
  evalf(t2);
  w2:=sqrt((x+y)^2+z^2);
  evalf(w2);
```

$$t2 := \frac{225\sqrt{5}}{14}$$
$$35.93680677$$
$$w2 := 50\sqrt{5}$$
$$111.8033988$$

(3) R rennt zuerst schräg zum Wasser - also zum Punkt S - um dann im Wasser die kürzeste Strecke zu P zu schwimmen.

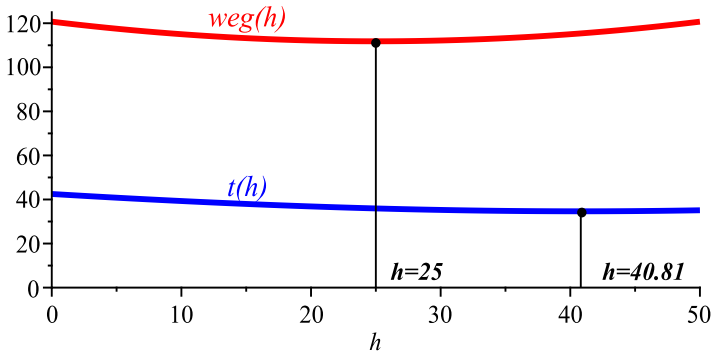
```
> t3:=sqrt(x^2+z^2)/vu+y/vw;
  evalf(t3);
  w3:=sqrt(x^2+z^2)+y;
  evalf(w3);
```

$$t3 := 25 + \frac{50\sqrt{2}}{7}$$
$$35.10152544$$
$$w3 := 50 + 50\sqrt{2}$$
$$120.7106781$$

(4) Allgemeine Lösung
R nimmt den Weg RQP. Er rennt zuerst schräg zum Wasser - also zum Punkt Q - um dann im Wasser zu P zu schwimmen.

```
> # Gesamtzeit als Funktion vom Abstand h=QH=0..z
  t:=unapply(sqrt(x^2+h^2)/vu+sqrt(y^2+(z-h)^2)/vw,h);
  # Weg als Funktion vom Abstand h
  weg:=unapply(sqrt(x^2+h^2)+sqrt(y^2+(z-h)^2),h);

> p10:=plot([t(h),weg(h)],h=0..z,0..125,color=[blue,red],
            thickness=3,labels=[`h`,``]):
  p11:=plot({[[40.81,0],[40.81,34.64]],[[25,0],[25,111.8]]},color=black):
  p12:=pointplot([[40.81,34.64],[25,111.8]],
                 symbol=solidcircle,symbolsize=16,color=black):
  p13:=textplot([[15,46,` t(h)`]],font=[TIMES,ROMAN,12],color=blue):
  p14:=textplot([[15,123,` weg(h)`]],font=[TIMES,ROMAN,12],color=red):
  p15:=textplot([[28,7,` h=25`],[45.5,7,` h=40.81`]],
                font=[TIMES,BOLD,10],color=black):

> display(p10,p11,p12,p13,p14,p15);
```

Die Funktionsverläufe von Zeit $t(h)$ und Weg $weg(h)$ zeigen, dass ihre minimalen Werte jeweils zu anderen Größen des Parameters h auftreten.

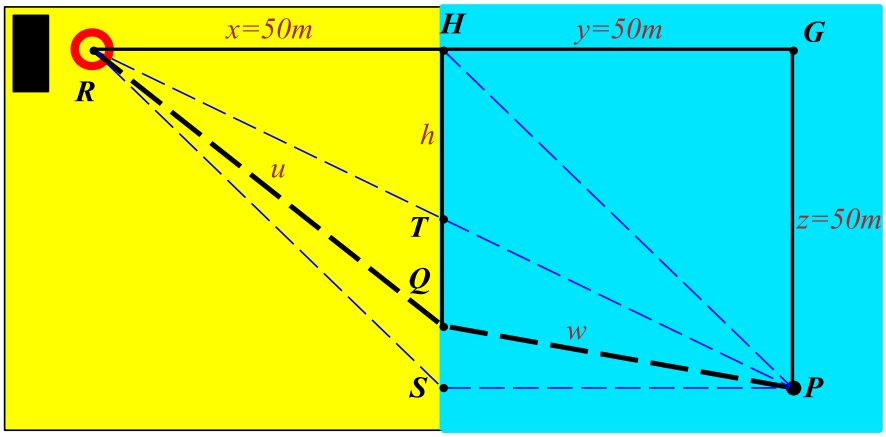


Die wegoptimale Situation ist im Punkt (2) beschrieben und gehört zum Parameterwert $h = 25$.
Die Berechnung der minimalen Zeit und dazu den Weg ergibt sich aus der Extremwertaufgabe $t(h) \to \min$ bzw. der dazugehörigen notwendigen Bedingung

$$t'(h) = \frac{1}{7}\frac{h}{\sqrt{2500 + h^2}} + \frac{1}{4}\frac{-100 + 2h}{\sqrt{5000 - 100h + h^2}} = 0.$$

Die Lösung ist

$$h_4 = 40.81505946, \quad t(h_4) = 34.63882219, \quad weg(h_4) = 115.3801758.$$



Für eine Animation der Situation kann man diese Strandaufnahme verwenden.
Zum Argument $h \in [0, 50]$ bieten sich 51 Frames an. Dazu lässt man im Film die Zahlenwerte h, $t(h)$ und $weg(h)$ mitlaufen.

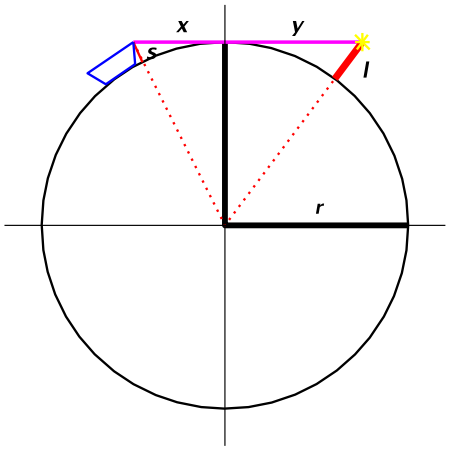
Hinter der Situation des Rettungsschwimmers verbirgt sich das **Prinzip Pierre de Fermat** (1608-1665) über das Verhalten von Licht.
Das Licht verhält sich wie ein perfekter Rettungsschwimmer, denn ein Lichtstrahl verläuft von einem Punkt A zu einem Punkt B stets auf dem zeitlich kürzesten Weg, egal welche Medien es dabei durchdringt.
Dabei ist das Licht nicht überall gleich schnell. So beträgt die Lichtgeschwindigkeit in Luft ca. $300\,000\,m/s$, in Wasser $225\,000\,m/s$ und in Glas $200\,000\,m/s$.

2.3.4 Entfernung des Schiffes vom Leuchtturm

Verzeichnis: `verschiedenes`
Datei: zeige_mal1.mws

Welche Entfernung hat das Schiff vom Leuchtturm, wenn es diesen erstmalig sieht? Aus einem vereinfachten Modell kann man schnell die Lösung berechnen.

Äquatorradius $r = 6377,397\,km$
(polarer Radius $= 6356,079\,km$)
Höhe des Leuchturms $l = 30\,m$
Höhe des Schiffes (Beobachtungsstelle) $s = 20\,m$




```
> # Groessenangaben in km
  r:=6377.397:
  l:=0.030:
  s:=0.020:
  x:=sqrt((r+s)^2-r^2):
  y:=sqrt((r+l)^2-r^2):
  Entfernung:=x+y;          # 35.533 km
```

2.3.5 Feuerwehrleiter

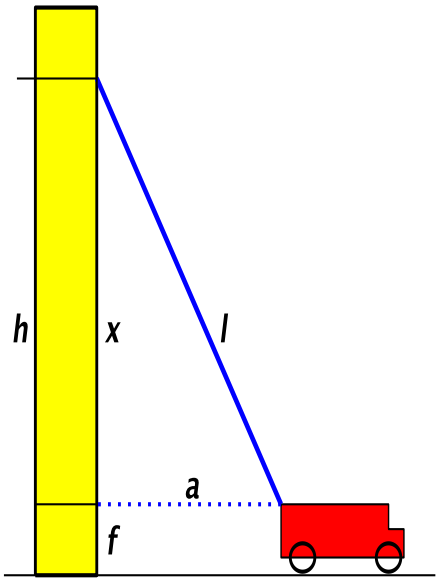
Verzeichnis: `verschiedenes`
Datei: zeige_mal1.mws

Ein Feuerwehrauto mit seiner $30\,m$ langen Feuerwehrleiter soll Personen aus den oberen Etagen eines Wohnblocks retten.
Der Mindestabstand des Autos vom Block beträgt $12\,m$.
Aus welcher Höhe können Personen über die Leiter absteigen?

Es gilt zunächst:

Mit dem kleinsten Abstand $a = 12\,m$ erreicht man die größte Höhe h.

Problemparameter und Situation



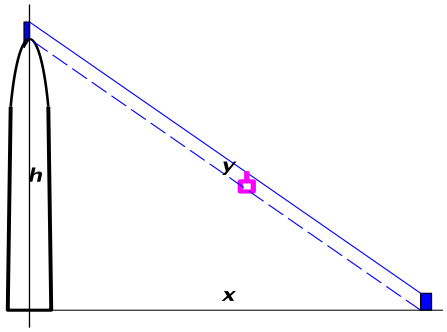

```
> f:=3;          # Fahrzeughoehe in m
  a:=12;         # Abstand in m
  l:=30;         # Laenge der Leiter in m
  x:=sqrt(l^2-a^2);
  h:=f+x;        # 3+6*sqrt(21)=30.49545417
```

2.3.6 Zuckerhut und Gondel

Verzeichnis: `verschiedenes`
Datei: zeige_mal1.mws

Eine Gondel braucht bei einer Geschwindigkeit von $v = 30\,km/h$ von der Talstation bis zur $180\,m$ hohen Spitze des Berges (Zuckerhut) 3 Minuten.
Wie weit ist die Talstation vom Berg (Fußpunkt der Bergspitze) entfernt?
Berechnungen mit einem vereinfachten Modell.

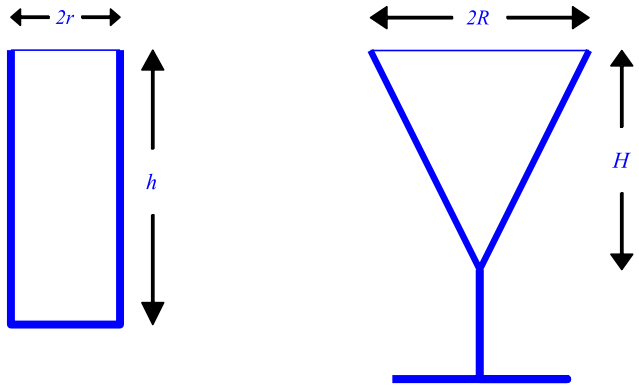
```
> # Gondelstrecke y
  v:=30:                 # km/h
  v:=v*1000/60:          # m/min
  t:=3:                  # min
  y:=v*t:                # m
  # Entfernung x
  h:=180:                # m
  # Pythagoras
  x:=sqrt(y^2-h^2);      # 120*sqrt(154)
                         # =1489.161 m
```



2.3.7 Sektglas

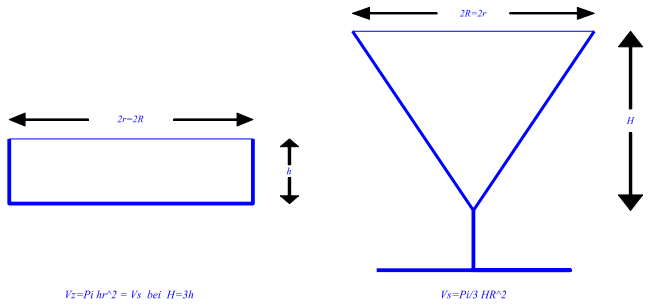
Verzeichnis: `sektglas`
Datei: sektglas1.mws
Modelle: Sektglas, Glas, Flüssigkeit

Es geht um das Umfüllen von Flüssigkeit von einem normalen Glas in ein Sektglas. Bei der Berechnung der Volumina gehen wir beim Glas von einem Zylinder sowie beim Sektglas von der Form eines Kreiskegels aus.
Wir stellen zunächst die Gläser dar, dann Füllungen und führen schließlich die Animation des Umfüllens für volumengleiche Gläser durch.



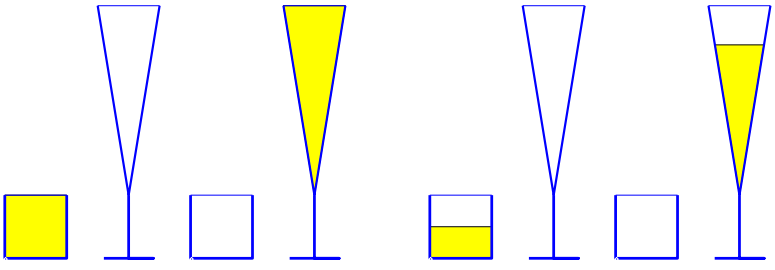
Damit nun bei gleichen Öffnungsradien der Gläser auch die Volumina übereinstimmen, muss für die "effektiven Höhen" die Beziehung $H = 3h$ gelten.

Volumengleiche Gläser, aber etwas unförmige Gestalt

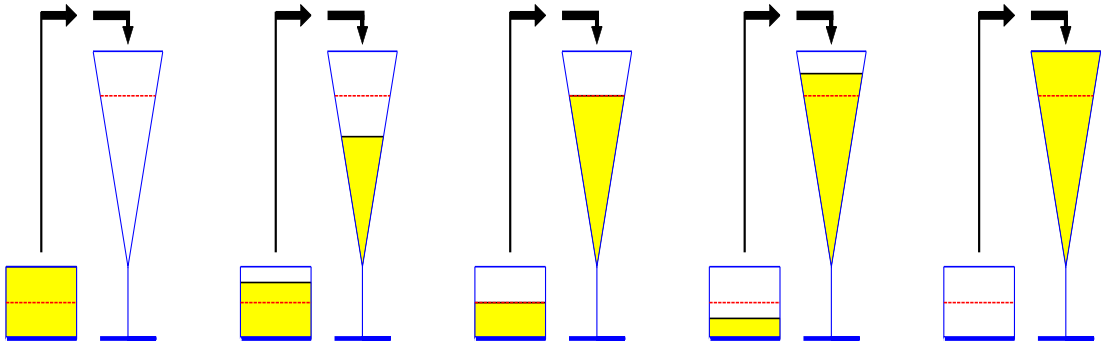



```
> setoptions(scaling=unconstrained,axes=none):
  pk:=vector(2,[]):
  s1:=plot([[0.2,0],[1.8,0],[1,0],[1,1]],color=blue,thickness=5):
  s2:=plot([[0,4],[1,1],[2,4]],color=blue,thickness=4):
  s3:=plot([[0,4],[2,4]],color=blue,thickness=1):
  ts1:=plottools[arrow]([1.3,4.3],[2,4.3],0.02,0.2,0.2,color=black):
  ts2:=plottools[arrow]([0.7,4.3],[0,4.3],0.02,0.2,0.2,color=black):
  ts3:=textplot([[1,4.3,`2R=2r`]],font=[TIMES,ROMAN,10],color=blue):
  ts4:=plottools[arrow]([2.3,2.3],[2.3,1],0.02,0.2,0.2,color=black):
  ts5:=plottools[arrow]([2.3,2.7],[2.3,4],0.02,0.2,0.2,color=black):
  ts6:=textplot([[2.3,2.5,`H`]],font=[TIMES,ROMAN,10],color=blue):
  ts7:=textplot([[1,-0.4,`Vs=Pi/3 HR^2`]],
                font=[TIMES,ROMAN,12],color=blue):

> z1:=plot([[0,2],[0,1],[2,1],[2,2]],color=blue,thickness=5):
  z2:=plot([[0,2],[2,2]],color=blue,thickness=1):
  z3:=plot([[0,0],[2,4]],style=point,color=white):
  tz1:=plottools[arrow]([1.35,2.3],[2,2.3],0.02,0.2,0.2,color=black):
  tz2:=plottools[arrow]([0.65,2.3],[0,2.3],0.02,0.2,0.2,color=black):
  tz3:=textplot([[1,2.3,`2r=2R`]],font=[TIMES,ROMAN,10],color=blue):
  tz4:=plottools[arrow]([2.3,1.4],[2.3,1],0.02,0.15,0.3,color=black):
  tz5:=plottools[arrow]([2.3,1.6],[2.3,2],0.02,0.15,0.3,color=black):
  tz6:=textplot([[2.3,1.5,`h`]],font=[TIMES,ROMAN,10],color=blue):
  tz7:=textplot([[1,-0.4,`Vz=Pi hr^2 = Vs  bei  H=3h`]],
                font=[TIMES,ROMAN,12],color=blue):

> pk[1]:=display(s1,s2,s3,ts1,ts2,ts3,ts4,ts5,ts6,ts7):
  pk[2]:=display(z1,z2,z3,tz1,tz2,tz3,tz4,tz5,tz6,tz7):
  display(pk[1]); display(pk[2]); display(pk);
```

Volumengleiche Gläser, voll bzw. halbvoll



Animation des Umfüllens und Dateiausgabe



```
> setoptions(scaling=unconstrained,axes=none):
  n:=50:
  pka:=array(0..n,[]):
  pk:=vector(2,[]):

> z4:=plot([[0,1/2],[2,1/2]],color=red,linestyle=3):
  s4:=plot([[0.21,3.38],[1.79,3.38]],color=red,linestyle=3):
  sz5:=plot([[0,0],[2,5]],style=point,color=white):
  tz1:=plot([[1,1.2],[1,4.55]],color=black,thickness=10):
  tz2:=plottools[arrow]([1,4.5],[2,4.5],0.1,0.3,0.3,color=black):
  tz3:=plot([[0,4.5],[1.06,4.5]],color=black,thickness=10):
  tz4:=plottools[arrow]([1,4.5],[1,4.1],0.1,0.3,0.6,color=black):

  for i from 0 to n do
   h:=(n-i)/n;
   rs:=(i/n)^(1/3);
   Hs:=3*rs;
   fz2:=polygon([[0,h],[0,0],[2,0],[2,h]],color=yellow):
   fs2:=polygon([[1-rs,1+Hs],[1,1],[1+rs,1+Hs],[1-rs,1+Hs]],
                color=yellow):
   pk[1]:=display(z1,z2,z3,z4,sz5,tz1,tz2,fz2):
   pk[2]:=display(s1,s2,s3,s4,sz5,tz3,tz4,fs2):
   pka[i]:=display(pk)
  end do:

> umfuellen:=[seq(pka[i],i=0..n)]:
  display(umfuellen,insequence=true,scaling=unconstrained);
```

GIF Animated Transparent File

```
> pfad:=...: name:=...:
  datei:=cat(pfad,name);
> interface(plotdevice=gif,
            plotoutput=datei,
            plotoptions=`width=200,height=200,
            transparent=true`);
  plots[display](umfuellen,insequence=true,scaling=unconstrained);
  interface(plotdevice=default);
```

2.4 Knobeln und Raten

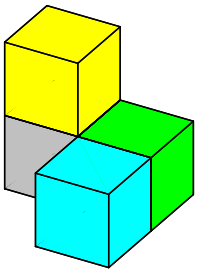
Natürlich sind auch hier mathematische Betrachtungen und Rechnungen sowie geometrische Darstellungen angebracht.

2.4.1 Anzahl der Würfel im Bauwerk

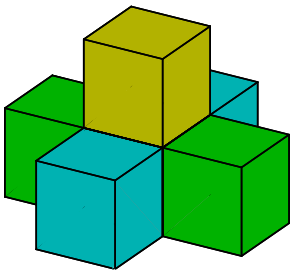
Verzeichnis: `bauwerk_aus_wuerfeln`
Dateien: cube1.mws, zeige_mal1.mws

Wie viele Würfel sind zum Bau der folgenden 3 Bauwerke benutzt worden?
Beschreibe, wie du beim 3. Bauwerk auf die Zahl der Bauklötze gekommen bist.

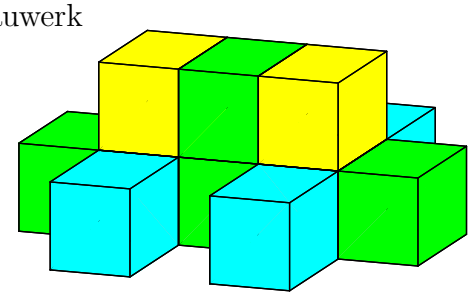
1. Bauwerk



2. Bauwerk



3. Bauwerk



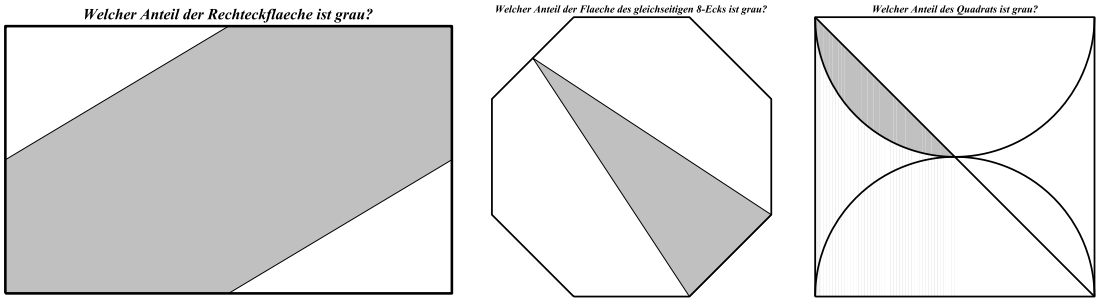
Konstruktion des 3. Bauwerks

```
> w1:=cuboid([0,0,0],[1,1,1],color=gray):
  w2:=cuboid([0,0,1],[1,1,2],color=yellow):
  w3:=cuboid([0,1,0],[1,2,1],color=green):
  w4:=cuboid([0,-1,0],[1,0,1],color=green):
  w5:=cuboid([1,0,0],[2,1,1],color=cyan):
  w6:=cuboid([-1,0,0],[0,1,1],color=cyan):
  v1:=cuboid([0,2,0],[1,3,1],color=gray):
  v2:=cuboid([0,2,1],[1,3,2],color=yellow):
  v3:=cuboid([0,3,0],[1,4,1],color=green):
  v4:=cuboid([0,1,1],[1,2,2],color=green):
  v5:=cuboid([1,2,0],[2,3,1],color=cyan):
  v6:=cuboid([-1,2,0],[0,3,1],color=cyan):

> plots[display]([w1,w2,w3,w4,w5,w6,v1,v2,v3,v4,v5,v6],
                 orientation=[20,60]);
```

2.4.2 Anteil der gefärbten Flaeche zur Gesamtfläche

Verzeichnis: `verschiedenes`
Datei: zeige_mal1.mws

Wieviel Prozent der Fläche ist grau eingefärbt?



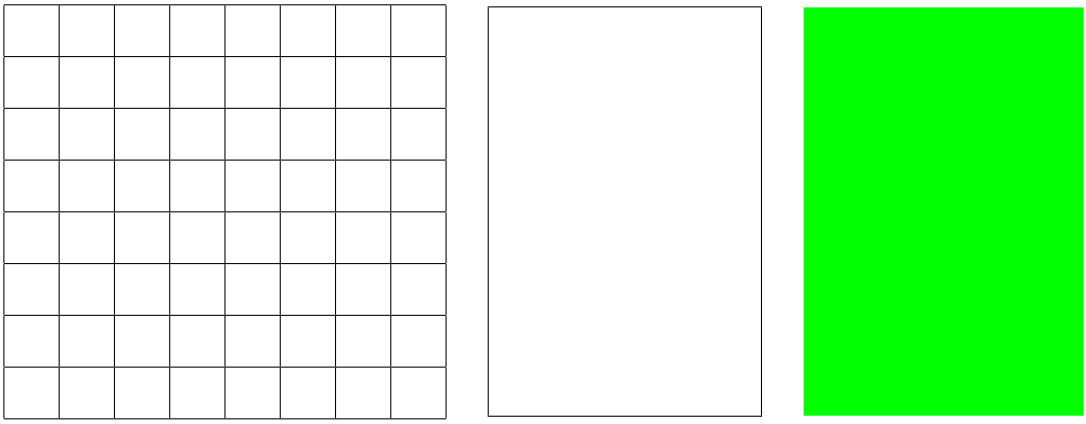

```
> # Rechteck
  p1:=plot([[0,0],[2,0],[2,1],[0,1],[0,0]],
           color=black,scaling=unconstrained):
  # Polygon (75%)
  p2:=polygon([[0,0],[1,0],[2,1/2],[2,1],[1,1],[0,1/2]],
              color=gray):

> #------------------------------------------------------------------
  # gleichseitiges 8-Eeck
  p1:=plot([[1/sqrt(2),0],[1+1/sqrt(2),0],[1+2/sqrt(2),1/sqrt(2)],
           [1+2/sqrt(2),1+1/sqrt(2)],[1+1/sqrt(2),1+2/sqrt(2)],
           [1/sqrt(2),1+2/sqrt(2)],[0,1+1/sqrt(2)],[0,1/sqrt(2)],
           [1/sqrt(2),0]],
           color=black,scaling=constrained):
  # Dreieck (25%)
  p2:=polygon([[1+1/sqrt(2),0],[1+2/sqrt(2),1/sqrt(2)],
              [1/(2*sqrt(2)),1+3/(2*sqrt(2))]],color=gray):

> #------------------------------------------------------------------
  # Quadrat
  p1:=plot([[0,0],[2,0],[2,2],[0,2],[0,0]],
           color=black,scaling=constrained):
  p2:=plot([sqrt(1-(x-1)^2),2-sqrt(1-(x-1)^2),2-x],x=0..2,
           color=black):
  # Kreisabschnitt (7.13%)
  p3:=plot(2-x,x=0..1,color=gray,filled=true):
  p4:=plot(2-sqrt(1-(x-1)^2),x=0..1,color=white,filled=true):

> # Rechnung zur letzten Situation
  # G:F = (Pi-2)/16:1, Anteil=7.13%
  b:=2:      # Seite des Quadrats
  r:=b/2:
  F:=b^2:
  G:=Pi*r^2/4-r*r/2:
  G/F*100;  evalf(%);
```

2.5 Basteln, Binden, Falten und Entfalten

2.5.1 Schachbrett, Bastelbogen und einfache Vielecke

Verzeichnis: `schachbrett_1gleich0`
Datei: schachbrettzerl2.mw
Modelle: Schachbrett, A4-Bogen, Bastelbogen

Was brauchen wir dafür? Eigentlich nur einen Bogen Papier.



Aber man nimmt meistens schon entweder einen quadratischen Bogen, einen Rechteckstreifen, ein kariertes A4-Blatt oder einfach ein A4-Blatt. Wichtig ist die Konstruktion/Falten von Grundfiguren, dazu gehören auch die regelmäßigen Vielecke/ Polygone.

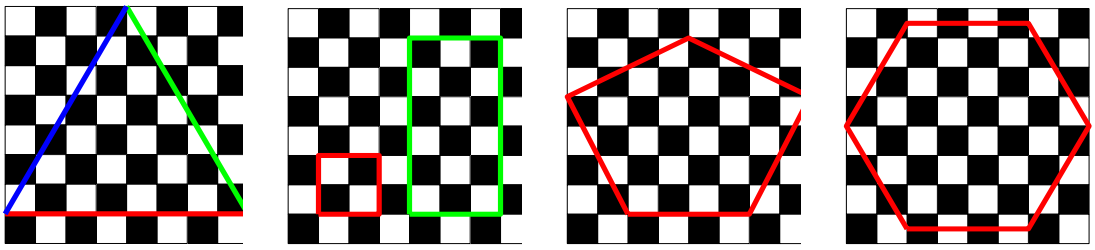
- Gleichseitiges Dreieck 

- Quadrat  , Rechteck 

- Fünfeck/Pentagon 

- Sechseck/Hexagon 

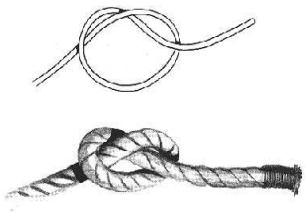
Näherungsweise Konstruktion der Polygone auf einem Schachbrett
Wie genau sind die Näherungen?

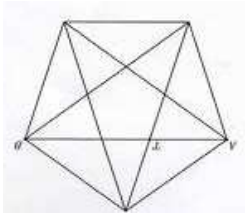


2.5.2 Seil und Fünfeck

Verzeichnisse: `knoten, entfalten_von_koerpern`
Dateien: knoten_einfach1.mws, Bastelbogen_5platon.pdf,
Bastelbogen__Dodekaeder.pdf, Bastelbogen__Minimalflaeche.pdf
Modelle: A4-Blatt, Bastelbogen, Papierstreifen, Seil

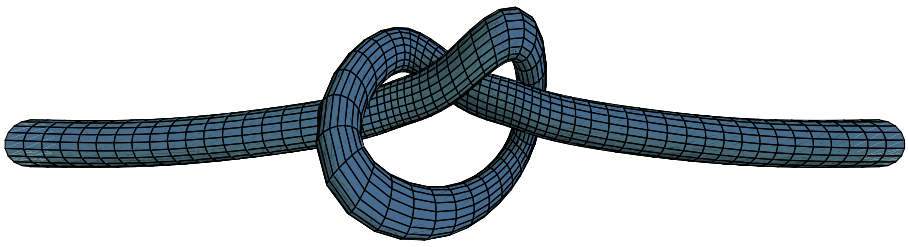
Fünfeck/Pentagon mittels Knoten
Einfacher Knoten/Überhandknoten mit Seil, Animation





Wie kompliziert ist es, den Überhandknoten zu konstruieren, oder anders gefragt wieviel Punkte braucht man, um den Knoten schön zu modellieren?
Der Knotenverlauf ist wie die Beschreibung einer Raumkurve $\mathcal{K}(t) = (x(t), y(t), z(t))$.
Bei Rechnungen in Maple habe ich dazu 17 signifikante Punkte im Raum verwendet, die für den Verlauf "wichtig" sind, und diese durch eine glatte Kurve verbunden.
Die mathematische Betrachtung basiert auf kubischen Splines.

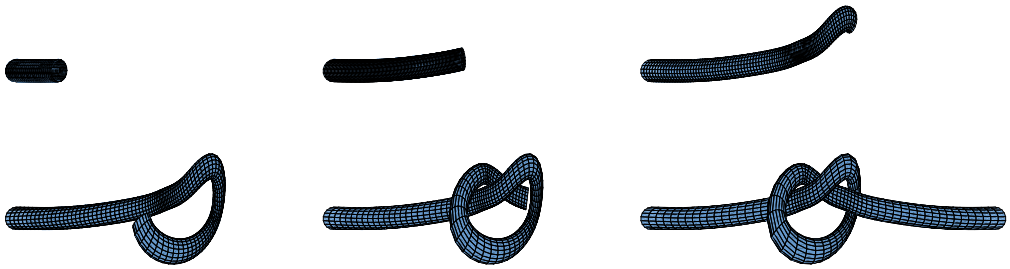
```
> tt:=[seq(i,i=0..16)]:
  xx:=[-4,-3,-2,-1,-0.5,0,0.6,0.8,0,-0.8,-0.6,0,0.5,1,2,3,4]:
  yy:=[0,0,0,0,-0.01,-0.15,-0.15,0.6,0,-0.6,0.15,0.15,0.01,0,0,0,0]:
  zz:=[0,0,0.05,0.15,0.25,0.40,0.6,0,-0.4,0,0.6,0.40,0.25,0.15,0.05,0,0]:
  t:='t':
  X:=spline(tt,xx,t):
  Y:=spline(tt,yy,t):
  Z:=spline(tt,zz,t):

> # Darstellung des Knotens
  Knoten_einfach:=plots[tubeplot]([X(t),Y(t),Z(t)],t=0..16,
    radius=1/7,tubepoints=20,numpoints=100,orientation=[-90,90],
    color=[0.4,0.6,0.8],projection=0.5,style=patch,scaling=unconstrained,
    ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(Knoten_einfach);
```



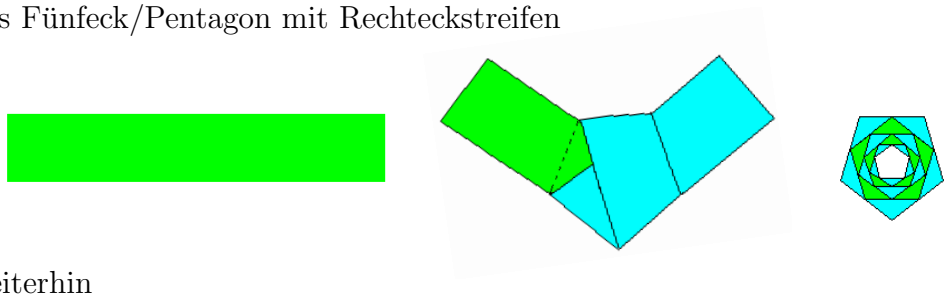
Die Animation des Knotens entspricht dem kontinuierlichen Wachstum des Seils.

```
> # Animation der Knotenentstehung
  t:='t':
  Knoten_einfach_Ani:=i->plots[tubeplot]([X(t),Y(t),Z(t)],t=0..16*i/n,
    radius=1/7,tubepoints=20,numpoints=100,orientation=[-90,90],
    color=[0.4,0.6,0.8],projection=0.5,style=patch,scaling=unconstrained,
    ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  n:=16:
  Knoten_Animation:=[seq(Knoten_einfach_Ani(i),i=1..n)]:
  display(Knoten_Animation,insequence=true,style=patch);
```

Von den 17 Frames sollen 6 angezeigt werden.

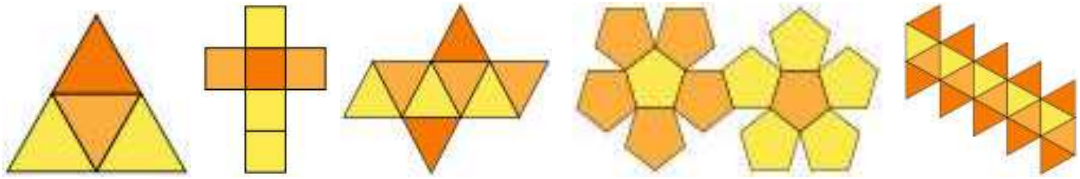


Das Fünfeck/Pentagon mit Rechteckstreifen



Weiterhin

- Siebzehneck (Heptadekagon)
  Das regelmäßige Siebzehneck ist unter alleiniger Verwendung von Zirkel und Lineal (Euklidische Werkzeuge) konstruierbar.
  Dies wurde von **Carl Friedrich Gauß** im Jahre 1796 nachgewiesen.
- Bastelbögen zu den 5 platonischen Körpern (**Platon** 428-348 v.u.Z.)



Tetraeder Hexaeder Oktaeder Dodekaeder Ikosaeder

- Minimalfläche, archimedische Körper (**Archimedes** 287-212 v.u.Z.)
- Die 5 platonischen Körper
  Unterschied zwischen platonischen und archimedischen Körpern

2.5.3 Knoten, Binden, Konstruktion, Animation

Verzeichnis: `knoten`
Dateien: knoten_einfach1.mws, palstek.mws, trossenstek1,2.mws, affenfaust.mws, zeige_mal5.mws, zeige_mal8.mws
Modelle: Seile, Leinen, Trossen, Affenfaust

- Knoten allgemein
  Nützlichkeit und Anwendung für regelmäßiges Fünfeck
  Man spricht vom Binden, Knüpfen oder Stecken von Knoten.
- Henkersknoten




Seemannsknoten und Eigenschaften

Von den zahlreichen Knoten sollen nur einige vorgestellt werden.

Palstek (Pfahlstich)
Der Palstek dient zum Knüpfen einer festen Schlaufe. Er ist der in der Seefahrt am häufigsten verwendete Knoten und wurde als König der Knoten bezeichnet.
Der Palstek zieht sich auch unter höchster Belastung nicht zu. Er lässt sich immer leicht wieder öffnen, denn er bekneift sich gut, ohne sich festzuziehen, und das machen nicht alle Knoten so.
Er dient zum Festmachen am Poller oder Pfahl und zum Retten von Personen.

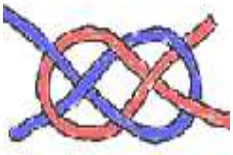
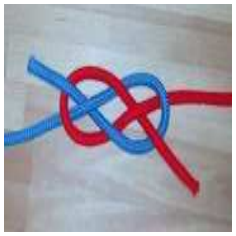




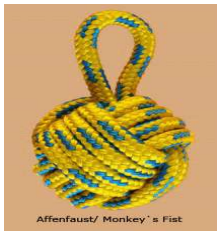
Kreuzknoten (Reffknoten), Unterschied zum Trossensteck



Trossenstek (Brezelknoten)
Dieser Knoten hat schon immer durch seine Schönheit Aufmerksamkeit erregt. Er besteht aus zwei ineinander verschlungene Augen und besitzt eine hohe Stabilität. Beim Zuziehen wird er allerdings sehr sperrig.
Dieser Knoten ist geeignet zum Verbinden zweier dicker Leinen oder Trossen auch unterschiedlicher Dicke. Er ist selbst nach starkem Zug stets wieder lösbar.

Affenfaust (Kindskopf, Schmeissleinenknoten)
Die Affenfaust ist so beliebt, weil sie so gut aussieht. Sie dient zum Beschweren des Endes einer Wurfleine, zur Sicherung beim Klettern oder als Zierknoten.


Rechnungen und Animationen in Maple

Wie beim Überhandknoten ist auch hier von Interesse, welche räumliche Punkte zur Modellierung der Knoten gewählt werden.

Palstek

Wir stellen für den Palstek 4 Varianten vor. Es zeigt sich, dass man mit eher weniger Punkten und passenden Radien schöne Knoten binden kann. Wegen der guten Krümmungseigenschaften verwenden wir die kubischen Splines.

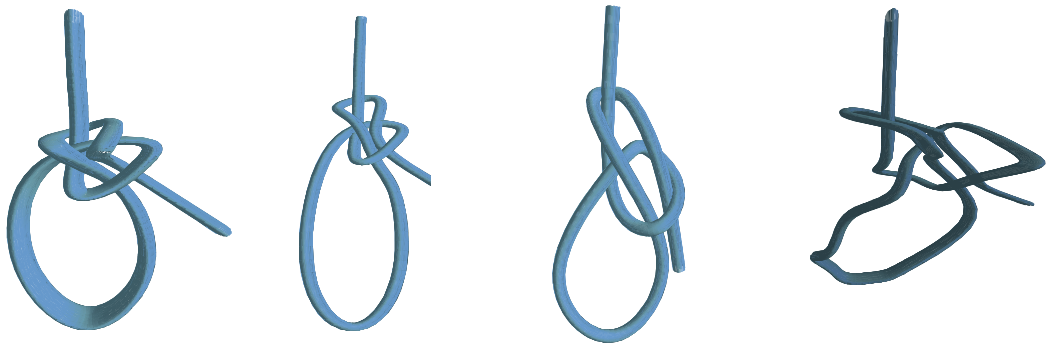
```
> # Variante 1
  m:=18:              # -> 19 Punkte
  tt:=[0,3/18,6/18,9/18,12/18,15/18,18/18,21/18,24/18,27/18,
       30/18,33/18,36/18,39/18,42/18,45/18,48/18,51/18,31/10]:
  xx:=[8,8,8,8,9,10,10,7,6,8,10,10,9,8,7.5,9,9,9,13]:
  xx:=xx*(-1):
  yy:=[2,5,8,10,10,9,8,9,14,17,15,12,10,8,7.5,7,8,9,13]:
  yy:=yy*2:
  zz:=[0,0,0,-1,-1,-1,-1,-1,0,0,0,0.5,0.5,-3,1,1,-3,0.5,0.5]:
  zz:=zz*(-0.5):

> t:='t':
  X:=spline(tt,xx,t):
  Y:=spline(tt,yy,t):
  Z:=spline(tt,zz,t):
```

```
> # Darstellung des Knotens
  Palstek:=plots[tubeplot]([X(t),Y(t),Z(t)],t=0..3.1,
    radius=2/7,tubepoints=50,numpoints=190,orientation=[90,26],
    color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
    scaling=unconstrained,
    ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(Palstek);

> # Animation der Knotenentstehung
  t:='t':
  Palstek_Ani:=i->plots[tubeplot]([X(t),Y(t),Z(t)],t=0..3.1*i/n,
       radius=2/7,tubepoints=30,numpoints=190,orientation=[90,26],
       color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
       scaling=unconstrained,
       ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  n1:=19:
  n:=n1-1:      # 18
  Palstek_Animation:=[seq(Palstek_Ani(i),i=1..n)]:
  display(Palstek_Animation,insequence=true,style=patch);


> #------------------------------------------------------------------------
  # Variante 2
  m:=20:                # -> 21 Punkte
  tt:=[-6/18,-3/18,0,3/18,6/18,9/18,12/18,15/18,18/18,21/18,24/18,
       27/18,30/18,33/18,36/18,39/18,42/18,45/18,48/18,51/18,31/10]:
  xx:=[8,8,8,8,8,8,9,10,10,7,6,8,10,10,9,8,7,9,9,9.5,13]:
  xx:=xx*(-1):
  yy:=[0,1,2,5,8,10,10,9,8,9,15,18,16,12,10,8,6.5,6,8,9.5,12]:
  yy:=yy*2:
  zz:=[0,0,0,0,0,-1,-1.5,-1.5,-1,-.5,0,1,1.5,1,0,-2,.75,.75,-1.25,-1,.5]:
  zz:=zz*(-0.5):

> n1:=nops(tt):         # 21
  n:=n1-1:
  tmin:=tt[1]-1e-2:     # fuer Grafik
  tmax:=tt[n1]+1e-2:

> t:='t':
  X:=spline(tt,xx,t):
  Y:=spline(tt,yy,t):
  Z:=spline(tt,zz,t):

# Darstellung des Knotens
  Palstek:=plots[tubeplot]([X(t),Y(t),Z(t)],t=tmin..tmax,
       radius=2/7,tubepoints=30,numpoints=190,orientation=[90,26],
       color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
       scaling=unconstrained,
       ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(Palstek);
# Animation der Knotenentstehung s.o.
```

```
> #-----------------------------------------------------------------------
  # Variante 3
  m:=14:                # -> 15 Punkte
  tt:=[seq(i,i=0..m)]:
  xx:=[1,0,2,1,0.5,1,0,0,0,1.75,0.5,-0.25,0.5,1.25,0]:
  xx:=[seq(xx[i],i=1..m+1)]:
  yy:=[0,0,2,3.5,2,0,-3,0,3,0.5,-1,0,1,2.5,4]:
  yy:=[seq(yy[i],i=1..m+1)]:
  zz:=[18,9,7,9,11,10,4,0,4,10,13.5,14,13.5,10,4]:
  zz:=[seq(zz[i],i=1..m+1)]:

> n1:=nops(tt):         # 15
  n:=n1-1:
  tmin:=tt[1]-1e-2:     # fuer Grafik
  tmax:=tt[n1]+1e-2:

> t:='t':
  X:=spline(tt,xx,t):
  Y:=spline(tt,yy,t):
  Z:=spline(tt,zz,t):

> # Darstellung des Knotens
  Palstek:=plots[tubeplot]([X(t),Y(t),Z(t)],t=tmin..tmax,
       radius=2/6,tubepoints=30,numpoints=190,orientation=[90,26],
       color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
       scaling=constrained,
       ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(Palstek);
# Animation der Knotenentstehung s.o.
```

Die 4. Variante zeigt, wie unförmig der Palstek wird bei schlechter Wahl der Punkte.

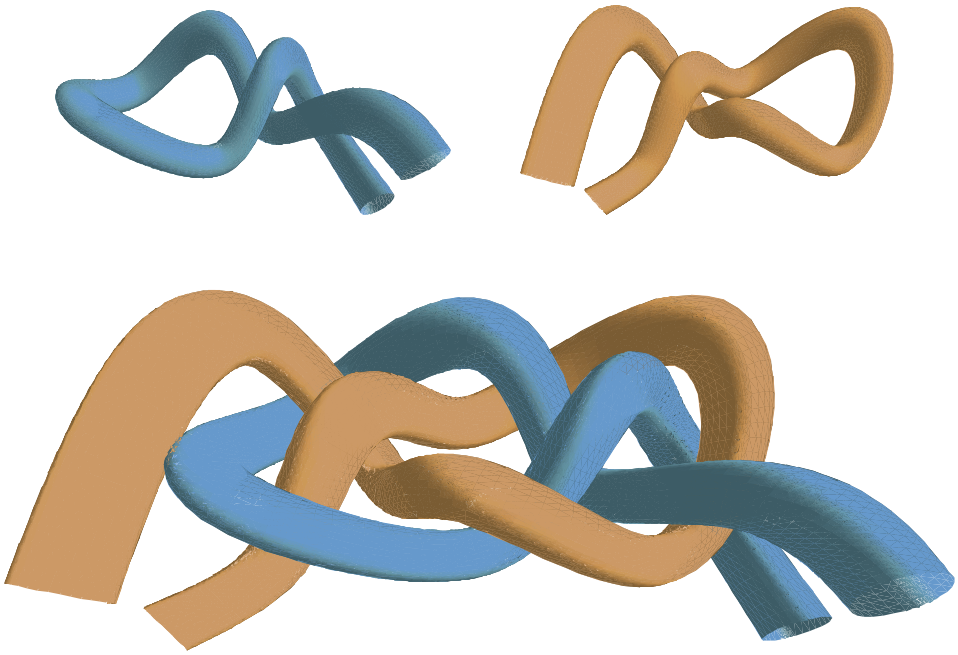
Variante 1 Variante 2 Variante 3 Variante 4 (33 Punkte)



Trossenstek

```
> tt:=[0,1/3,2/3,1,4/3,3/2,5/3,2,7/3,8/3,31/10];
  xx:=[-6,-4,-2,0,4,24/5,4,0,-2,-4,-6];
  yy:=[4,2,0,-2,-2,0,2,2,0,-2,4];
  zz:=[1,1,-1,1,-1,0,1,-1,1,-1,-1];
  n1:=nops(tt);       # 11
  n:=n1-1;
```

```
> t:='t':
  X:=spline(tt,xx,t);
  Y:=spline(tt,yy,t);
  Z:=spline(tt,zz,t);

> # 1.Seil
  p1:=plots[tubeplot]([X(t),Y(t),Z(t)],t=0..3.1,
        radius=4/7,tubepoints=30,numpoints=190,orientation=[90,26],
        color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
        scaling=unconstrained,
        ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(p1);

> # 2.Seil
  p2:=plots[tubeplot]([-X(t),Y(t),-Z(t)],t=0..3.1,
        radius=4/7,tubepoints=30,numpoints=190,orientation=[90,26],
        color=[0.8,0.6,0.4],projection=0.5,style=patchnogrid,
        scaling=unconstrained,
        ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(p2);

> display(p1,p2);
```



Bei der Animation beider Seile nacheinander sollte man auf die Dimensionen der Parameter achten, den bei guten grafische Auflösungen können die exportierten eps-Dateien zu den Bildern schnell sehr groß werden (über 10 MByte). Deshalb werden nachfolgend auch keine Frames der Animation angezeigt.

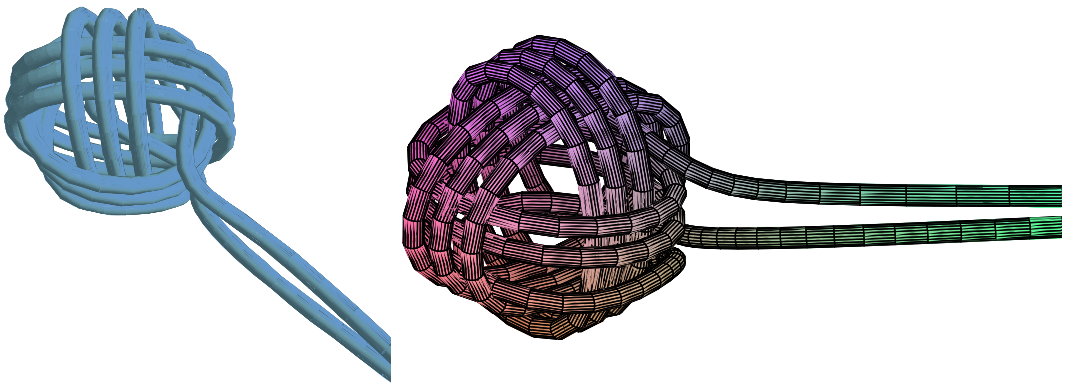
```
> # Animation
  curve1:=[X(t),Y(t),Z(t)]:
  d11:=i->plots[tubeplot](curve1,t=0..3.1*i/n,
            radius=4/7,tubepoints=20,numpoints=150,orientation=[90,26],
            color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
            scaling=unconstrained,
            ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  n:=8:
  dk11:=[seq(d11(i),i=1..n)]:

> curve2:=[-X(t),Y(t),-Z(t)]:
  d21:=i->plots[tubeplot](curve2,t=0..3.1*i/n,
            radius=4/7,tubepoints=20,numpoints=150,orientation=[90,26],
            color=[0.8,0.6,0.4],projection=0.5,style=patchnogrid,
            scaling=unconstrained,
            ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  dk21:=[seq(d21(i),i=1..n)]:

> dkh:=k->plots[display]([d11(n),seq(d21(i),i=1..k)]):
  dkh1:=plots[display](d11(n),d21(n)):
  dkg:=[seq(d11(i),i=1..n),d11(n),seq(dkh(k),k=1..n),dkh1]:
  plots[display](dkg,insequence=true);
```

Affenfaust

Für diesen Knoten braucht man viel Seil. Man kann ihn zweifach, aber auch dreifach und mehr binden. Hier knüpfen wir ihn dreifach. Zur Animation zeigen wir nur das Endbild. Für feine Auflösungen enstehen lange Rechenzeiten und große Dateien.
Beim Klettern wird die Affenfaust in der Sächsischen Schweiz als Klemmsicherung eingesetzt, da die Verwendung von metallischen Klemmkeilen dort nicht gestattet ist. Beim Poi-Schwingen werden Affenfäuste als runde Brennkörper für Feuerakrobatik benutzt.

```
> # Affenfaust-Punkte
  xx:=[-10,0,
       5,5,15,15,5,5,15,15,5,5,15,15,7,5,7,
       16,16,4,4,16,16,4,4,16,16,4,2.5,6,
       8,8,8,8,10,10,10,10,12,12,12,12,
       12,5,4,-10]:
  x:=xx*(-1):
  yy:=[-10,0,
       5,15,15,5,5,15,15,5,5,15,15,5,4,5.5,7,
       8,8,8,8,10,10,10,10,12,12,12,12,14,
       14,6,6,14,14,6,6,14,14,6,6,14,
       10,7,4,-10]:
  yy:=yy*(-1):
  zz:=[0,0,
       -2,-2,-2,-2,0,0,0,0,2,2,2,3,4,4.5,5,
       4,-4,-4,4,4,-4,-4,4,4,-4,-4,4,4,
       -6,-6,6,6,-6,-6,6,6,-6,-6,6,6,
       0,3,3,2]:
  zz:=zz*(-1):
```

```
> test_x:=nops(xx);     # 46
  l:=test_x-1:
  tt:=[seq(i*3,i=0..l-1),31*l/10]:
  tt:=tt/l:
  t:='t':
  test_t:=nops(tt);     # 46

> X:=spline(tt,xx,t):
  Y:=spline(tt,yy,t):
  Z:=spline(tt,zz,t):

> # Darstellung des Knotens
  Affenfaust:=plots[tubeplot]([X(t),Y(t),Z(t)],t=0..3.1,
      radius=4/7,tubepoints=50,numpoints=250,orientation=[90,26],
      color=[0.4,0.6,0.8],projection=0.5,style=patchnogrid,
      scaling=constrained,
      ambientlight=[0.6,0.6,0.5],light=[75,50,1,0.9,0.7]):
  display(Affenfaust);

> # Animation der Knotenentstehung
  t:='t':
  n:=50:
  Affenfaust_Ani:=i->plots[tubeplot]([X(t),Y(t),Z(t)],t=0..3.1*i/n,
        radius=4/7,tubepoints=21,numpoints=5*i,
        style=patch,scaling=unconstrained):
  Affenfaust_Animation:=[seq(Affenfaust_Ani(i),i=1..n)]:
  display(Affenfaust_Animation,
        insequence=true,style=patch,scaling=unconstrained);
```

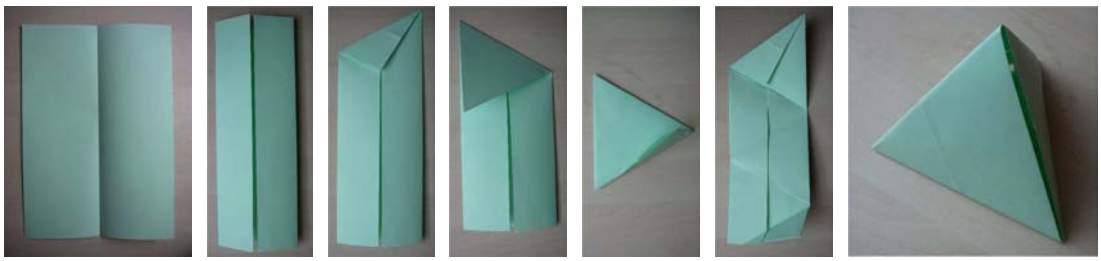


2.5.4 Archimedische Körper

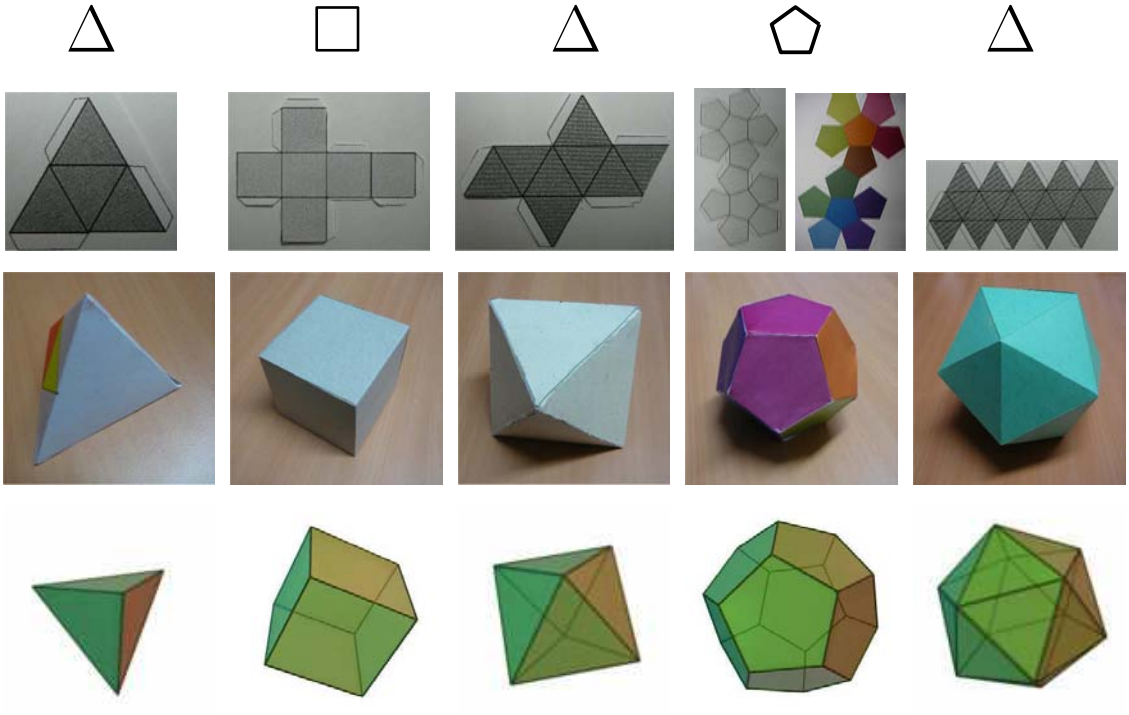
Verzeichnisse: `entfalten_von_koerpern, flaechen_3d`
Dateien: cube2.mws, zeige_mal7.mws, zeige_mal6.mws,
Bastelbogen_5platon.pdf, Bastelbogen_Tetraeder1.pdf,
Bastelbogen_Hexaeder1.pdf, Bastelbogen_Oktaeder1,2.pdf,
Bastelbogen_Dodekaeder1.pdf, Bastelbogen_Ikosaeder1,2,3.pdf,
Bastelbogen_Abgest_Ikosaeder1.pdf, PlatonischeKoerper.pdf,
Modelle: Polyeder, Bastelbogen, A4-Blatt

Die 5 platonischen Körper

Warum gibt es keine weiteren?
Der Eulersche Polyedersatz: #Ecken + #Flächen − #Kanten = 2
Konstruktion aus regelmäßigen Vielecken
Dreieck (#4) ⇒ Tetraeder, Falten des Tetraeders



Polygon/Grundfigur ⇒ Bastelbogen ⇒ Polyeder

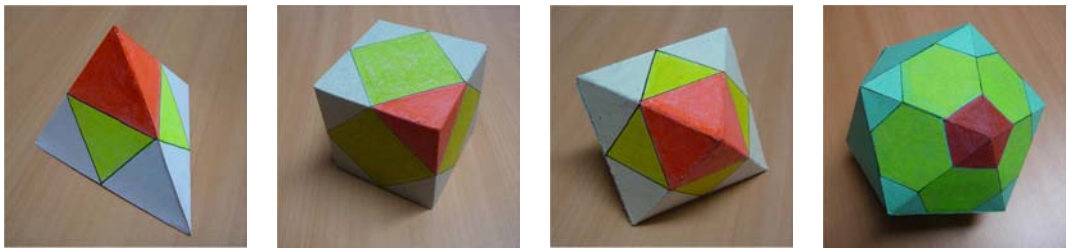


Quadrat (#6) ⇒ Hexaeder, Würfel, Kubus
Dreieck (#8) ⇒ Oktaeder, abgestumpfter Tetraeder
Fünfeck (#12) ⇒ Dodekaeder
Dreieck (#20) ⇒ Ikosaeder

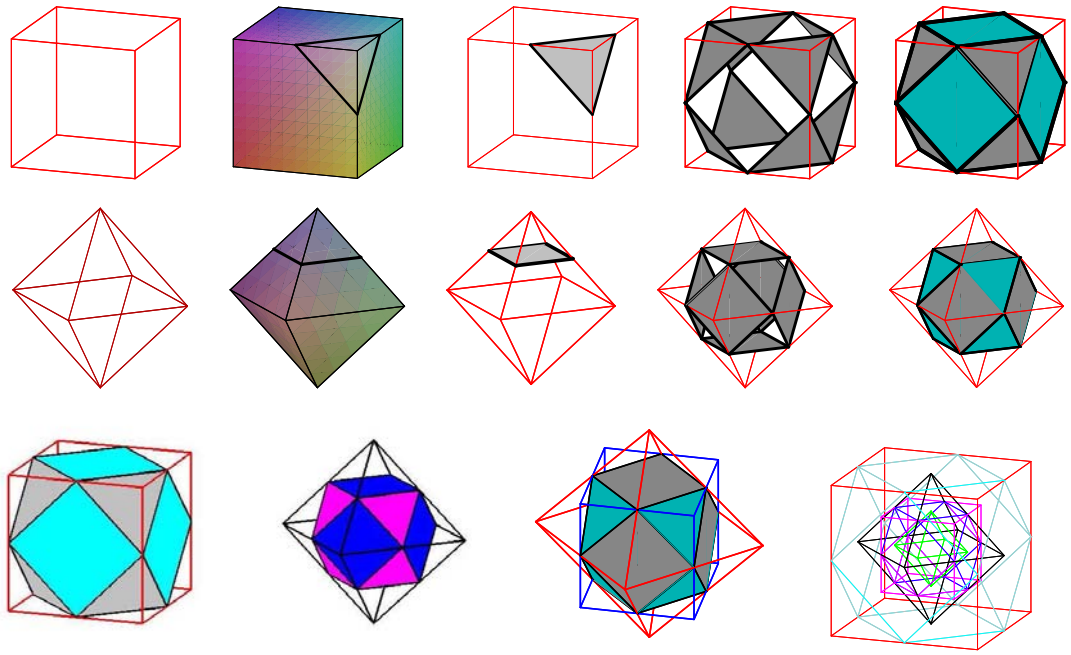
Abgestumpfte Archimedische Körper

Alle platonischen Körper können an den Ecken abgestumpft werden.

Einige Abstumpfungen:
Abgestumpfter Tetraeder ⇒ mit 4 Dreiecken, 4 Sechsecken
Abgestumpfter Tetraeder ⇒ Oktaeder
Abgestumpfter Hexaeder ⇒ mit 8 Dreiecken , 6 Achtecken
⇒ **kubischer Oktaeder** (8 Dreiecke, 6 Quadrate)
Abgestumpfter Oktaeder ⇒ kubischer Oktaeder
Abgestumpfter Oktaeder ⇒ mit 8 Sechsecken, 6 Quadraten
Abgestumpfter Ikosaeder ⇒ Ikosidodekaeder (12 Fünf-, 20 Dreiecke)
⇒ **Fußball** (12 Fünfecke, 20 Sechsecke)



Würfelstumpf (abgestumpfter Hexaeder), Dualität der Abstumpfung
Kubischer Oktaeder (abgestumpfter Oktaeder, Mittelkristall)



Maple-Anweisungen zu zwei Bilderserien mit dem kubischen Oktaeder

1. Bilderserie

```
> # Kubischer Oktaeder als abgestumpfter Hexaeder
  # 1.Figur
  f8 := hexahedron([0,0,0],1):
  f81:=plots[display](f8,style=wireframe,color=red,orientation=[20,75]):
  plots[display](f81);

  # 12 Kantenmittelknoten
  p9:=[[1,0,1],[0,1,1],[-1,0,1],[0,-1,1],[1,1,0],[1,-1,0],
       [-1,-1,0],[-1,1,0],[1,0,-1],[0,1,-1],[-1,0,-1],[0,-1,-1]]:
  # 8 Schnitte an den Ecken, woraus Dreiecke entstehen
  l91:=[p9[1],p9[2],p9[5],p9[1]]:
  l92:=[p9[6],p9[1],p9[4],p9[6]]:
  l93:=[p9[8],p9[3],p9[2],p9[8]]:
  l94:=[p9[7],p9[3],p9[4],p9[7]]:
  l95:=[p9[5],p9[10],p9[9],p9[5]]:
  l96:=[p9[6],p9[9],p9[12],p9[6]]:
  l97:=[p9[8],p9[11],p9[10],p9[8]]:
  l98:=[p9[7],p9[11],p9[12],p9[7]]:
  f91:=polygonplot3d(l91,color=gray,thickness=2,scaling=constrained):
  f92:=polygonplot3d(l92,color=gray,thickness=2,scaling=constrained):
  f93:=polygonplot3d(l93,color=gray,thickness=2,scaling=constrained):
  f94:=polygonplot3d(l94,color=gray,thickness=2,scaling=constrained):
  f95:=polygonplot3d(l95,color=gray,thickness=2,scaling=constrained):
  f96:=polygonplot3d(l96,color=gray,thickness=2,scaling=constrained):
  f97:=polygonplot3d(l97,color=gray,thickness=2,scaling=constrained):
  f98:=polygonplot3d(l98,color=gray,thickness=2,scaling=constrained):

  # 2.Figur
  plots[display](f8,f91,orientation=[20,75]);
  # 3.Figur
  plots[display]([f81,f91]);
  # 4.Figur
  plots[display](f81,f91,f92,f93,f94,f95,f96,f97,f98);

  # 6 Quadrate
  d91:=[p9[1],p9[2],p9[3],p9[4],p9[1]]:
  d92:=[p9[9],p9[10],p9[11],p9[12],p9[9]]:
  d93:=[p9[1],p9[5],p9[9],p9[6],p9[1]]:
  d94:=[p9[3],p9[8],p9[11],p9[7],p9[3]]:
  d95:=[p9[2],p9[5],p9[10],p9[8],p9[2]]:
  d96:=[p9[4],p9[6],p9[12],p9[7],p9[4]]:
  e1:=polygonplot3d(d91,color=cyan,thickness=2,scaling=constrained):
  e2:=polygonplot3d(d92,color=cyan,thickness=2,scaling=constrained):
  e3:=polygonplot3d(d93,color=cyan,thickness=2,scaling=constrained):
  e4:=polygonplot3d(d94,color=cyan,thickness=2,scaling=constrained):
  e5:=polygonplot3d(d95,color=cyan,thickness=2,scaling=constrained):
  e6:=polygonplot3d(d96,color=cyan,thickness=2,scaling=constrained):

  # 5.Figur
  plots[display](f81,f91,f92,f93,f94,f95,f96,f97,f98,e1,e2,e3,e4,e5,e6);
```

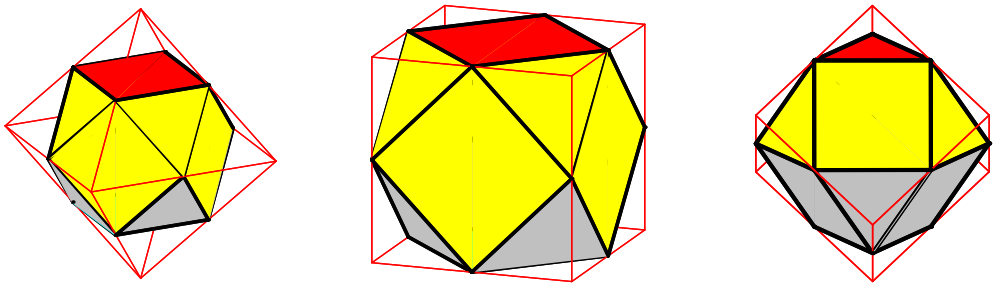
2. Bilderserie

```
> # Kubischer Oktaeder als abgestumpfter Oktaeder
  # 1.Figur
  f6 := octahedron([0,0,0],1):
  f61:=plots[display](f6,style=wireframe,color=red,orientation=[20,75]):
  plots[display](f61);

> # 8 Kantenmittelknoten
  p7:=[[1/2,0,1/2],[0,1/2,1/2],[-1/2,0,1/2],[0,-1/2,1/2],
       [1/2,1/2,0],[1/2,-1/2,0],[-1/2,-1/2,0],[-1/2,1/2,0],
       [1/2,0,-1/2],[0,1/2,-1/2],[-1/2,0,-1/2],[0,-1/2,-1/2]]:
  # 6 Schnitte an den Ecken, woraus Quadrate entstehen
  l71:=[p7[1],p7[2],p7[3],p7[4],p7[1]]:
  l72:=[p7[1],p7[5],p7[9],p7[6],p7[1]]:
  l73:=[p7[2],p7[5],p7[10],p7[8],p7[2]]:
  l74:=[p7[3],p7[7],p7[11],p7[8],p7[3]]:
  l75:=[p7[4],p7[6],p7[12],p7[7],p7[4]]:
  l76:=[p7[9],p7[10],p7[11],p7[12],p7[9]]:
  f71:=polygonplot3d(l71,color=gray,thickness=2,scaling=constrained):
  f72:=polygonplot3d(l72,color=gray,thickness=2,scaling=constrained):
  f73:=polygonplot3d(l73,color=gray,thickness=2,scaling=constrained):
  f74:=polygonplot3d(l74,color=gray,thickness=2,scaling=constrained):
  f75:=polygonplot3d(l75,color=gray,thickness=2,scaling=constrained):
  f76:=polygonplot3d(l76,color=gray,thickness=2,scaling=constrained):

  # 2.Figur
  plots[display](f6,f71,orientation=[20,75]);
  # 3.Figur
  plots[display](f61,f71);
  # 4.Figur
  plots[display](f61,f71,f72,f73,f74,f75,f76);

  # 8 Dreiecke
  d71:=[p7[1],p7[2],p7[5],p7[1]]:
  d72:=[p7[2],p7[3],p7[8],p7[2]]:
  d73:=[p7[3],p7[4],p7[7],p7[3]]:
  d74:=[p7[4],p7[1],p7[6],p7[4]]:
  d75:=[p7[9],p7[10],p7[5],p7[9]]:
  d76:=[p7[10],p7[11],p7[8],p7[10]]:
  d77:=[p7[11],p7[12],p7[7],p7[11]]:
  d78:=[p7[12],p7[9],p7[6],p7[12]]:
  d1:=polygonplot3d(d71,color=cyan,thickness=2,scaling=constrained):
  d2:=polygonplot3d(d72,color=cyan,thickness=2,scaling=constrained):
  d3:=polygonplot3d(d73,color=cyan,thickness=2,scaling=constrained):
  d4:=polygonplot3d(d74,color=cyan,thickness=2,scaling=constrained):
  d5:=polygonplot3d(d75,color=cyan,thickness=2,scaling=constrained):
  d6:=polygonplot3d(d76,color=cyan,thickness=2,scaling=constrained):
  d7:=polygonplot3d(d77,color=cyan,thickness=2,scaling=constrained):
  d8:=polygonplot3d(d78,color=cyan,thickness=2,scaling=constrained):

  # 5.Figur
  plots[display](f61,f71,f72,f73,f74,f75,f76,d1,d2,d3,d4,d5,d6,d7,d8);
```

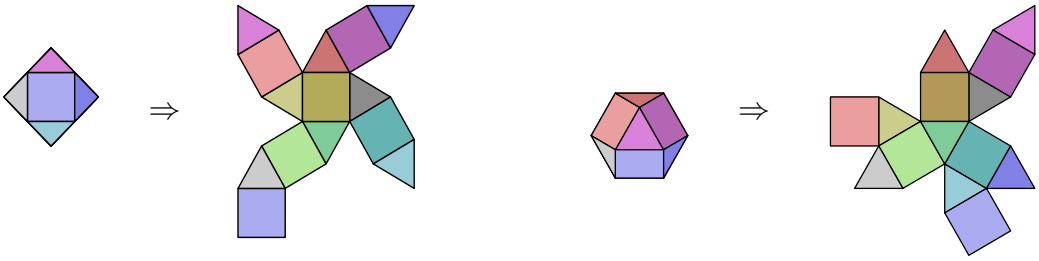
2.5.5 Entfaltung des kubischen Oktaeders

Verzeichnisse: `entfalten_von_koerpern, flaechen_3d`
Dateien: cube3.mws, zeige_mal6,7.mws, polyeder_farben1.mws,
Bastelbogen_KubOktaeder2.pdf
Modelle: Bastelbögen, Polyeder

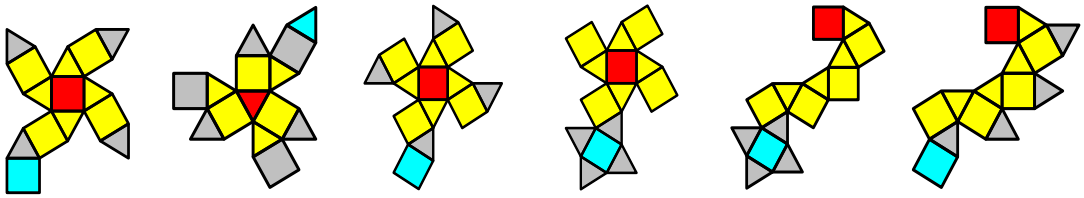
Die Konstruktion bzw. das Falten eines archimedischen Körpers aus dem Bastelbogen geht einher mit der Entfaltung dieses Körpers, denn damit entsteht ja erst seine Projektion in die Ebene, also der zugehörige Bastelbogen.
Zahlreiche Körper lassen sich schön falten und entfalten.
Als Musterbeispiel für die Entfaltung zeigen wir den kubischen Oktaeder.



Entfaltung bezüglich einer Grundfläche, nämlich mit Quadrat bzw. Dreieck.



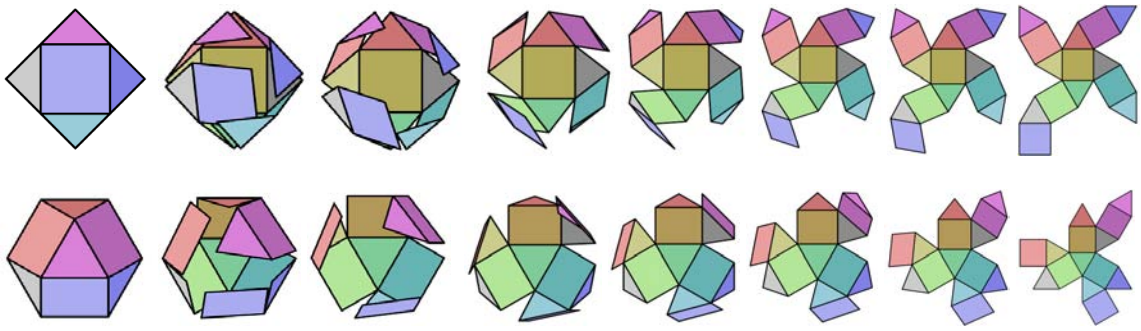
Übersicht zu den Entfaltungsvarianten V1-V6



Vergleich der rechteckigen Ausbreitung der Entfaltung der 6 Versionen
bei einer Kantenlänge 1 des Oktaeders

```
V1: [-1.5..2.5, -2.5..2.5] -> 4*5     = 20     -> kleinste Ausbreitung
V2: [-2..2.5,   -3..2.5  ] -> 4.5*5.5 = 24.75
V3: [-2..3,     -3..3    ] -> 5*6     = 30
V4: [-1.5..2.5, -3.5..2.5] -> 4*6     = 24
V5: [-1.5..4,   -3.5..2.5] -> 5.5*6   = 33
V6: [-1..4.5,   -3..2.5  ] -> 5.5*5.5 = 30.25
```

Frames zur Entfaltung für die Varianten 1 und 2
Die Dimensionen der Bilder werden v.l.n.r. verkleinert.



Die Entfaltung bedeutet die Animation mit Übergang der Ausgangssituation als Körper zum Endzustand als Bastelbogen. Dabei beschreiben die Ecken des Körpers eine Bahn. Diese sollte so gewählt werden, dass Ansicht der Zwischenstufen und Perspektive passend sind, keine Bahnüberschneidungen auftreten, Parallelität möglichst eingehalten wird, die Seitenflächen der Körper nicht "verstümmelt" werden, u.ä.
Die Bahnkurve vom einem Punkt P_k zu einem Punkt Q_k kann man im einfachsten Fall mittels der konvexen Linearkombination

$$Z_k \;=\; (1-t)\,P_k + t\,Q_k, \quad t \in [0,1],$$

beschreiben. Bei anderen Kurvenverläufen verwendet man z.B.

$$\begin{aligned} Z_k &= (1-t^m)\,P_k + t^m\,Q_k, \quad m = 2,4,6,8,..., \\ Z_k &= (1-t)^m\,P_k + \sqrt{t^l}\,Q_k, \quad m = 2,4,\ l = 3,..., \\ Z_k &= \cos(\tfrac{\pi}{2}t)\,P_k + (1-\cos(\tfrac{\pi}{2}t))\,Q_k, \end{aligned}$$

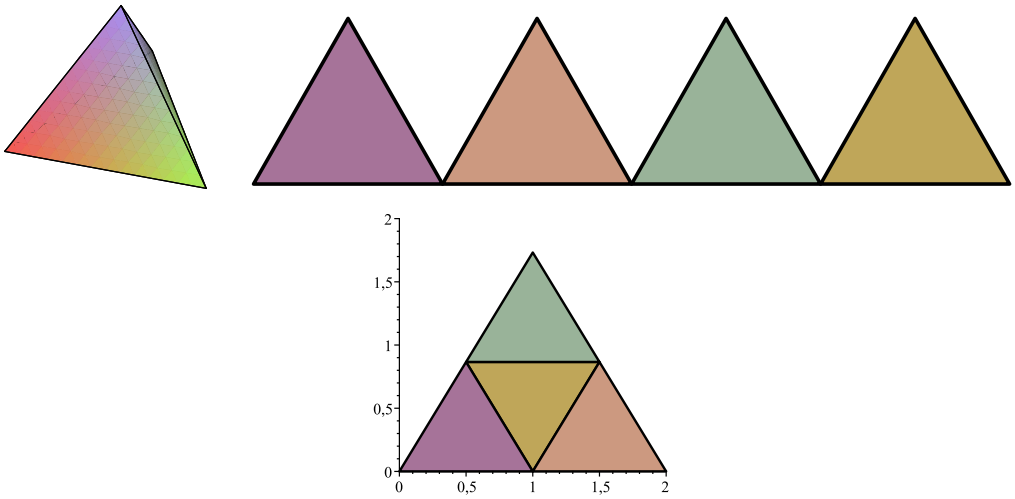
wobei $t \in [0,1]$. Bei der Entfaltung der platonischen und anderer Körper erfolgt die Auswahl aufgrund von Erfahrung und Fingerspitzengefühl.

Um die Entfaltung möglichst anschaulich zu machen, färbt man die Seiten des Polyeders unterschiedlich ein.
Es sollen die 5 Polyeder mit den ausgewählte Farbpaletten und einer ihrer zahlreichen Entfaltungen vorgestellt werden.

Tetraeder

```
> geom3d[tetrahedron](f4,point(o,0,0,0),1):
  draw(f4);
> f4:=plottools[tetrahedron]([0,0,0],1):
      # 1 = Entfernung von Mitte zu Kante
      # ra = sqrt(3), Radius der Aussenkugel
      # s = 2*sqrt(2), Kantenlaenge
      # h = s*sqrt(3)/2, Hoehe des Dreiecks
  display(f4,scaling=constrained);
```

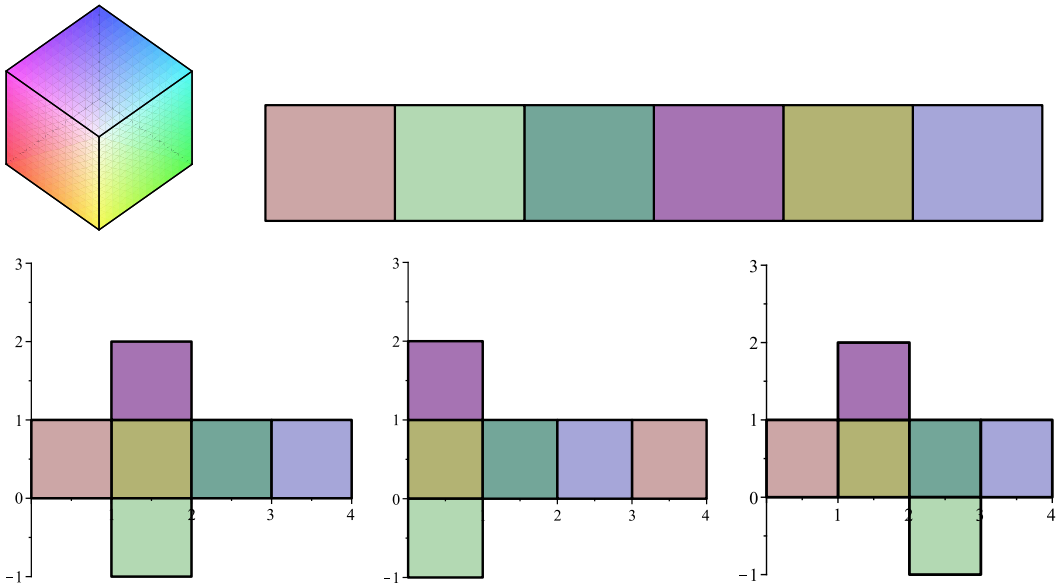
```
> # Prozedur fuer Dreieck
  p_t11:=proc(lu,s,farbe)    # Spitze oben
    local l,p12,r,h:         # lokale Parameter
    r:=s*sqrt(3)/3:          # Radius des Umkreises
    h:=s*sqrt(3)/2;          # Hoehe des Dreiecks
    l:=[lu,[lu[1]+s,lu[2]],[lu[1]+s/2,lu[2]+h],lu]:
    p12:=polygon(l,color=farbe,thickness=2):
    plots[display](p12,axes=none,scaling=constrained):
  end proc:

> # Farbpalette
  macro(farbe1=COLOR(RGB,0.65,0.45,0.60)):    # lila
  macro(farbe2=COLOR(RGB,0.80,0.60,0.50)):    # lachs
  macro(farbe3=COLOR(RGB,0.60,0.70,0.60)):    # cyan, dunkel
  macro(farbe4=COLOR(RGB,0.75,0.65,0.35)):    # oker

> display(p_t11([0,0],1,farbe1),p_t11([1,0],1,farbe2),
          p_t11([2,0],1,farbe3),p_t11([3,0],1,farbe4),
          scaling=unconstrained);
```



Hexaeder

```
> f6:=plottools[cuboid]([0,0,0],[1,1,1]):
  display(f6,scaling=constrained);
> geom3d[hexahedron](f6,point(o,0,0,0),sqrt(3)/2):
  draw(f6);
> f6:=plottools[hexahedron]([0,0,0],1):
      # ri = 1 = Radius der Innenkugel
      # ra = ri*sqrt(3) = sqrt(3), Radius der Aussenkugel
      # s = 2*ri = 2, Kantenlaenge
  display(f6,scaling=constrained);

> # Prozedur fuer Quadrat
> p_h1:=proc(lu,s,farbe)
    local l,p12,r:           # lokale Parameter
    r:=s/sqrt(2):            # Radius des Umkreises
```

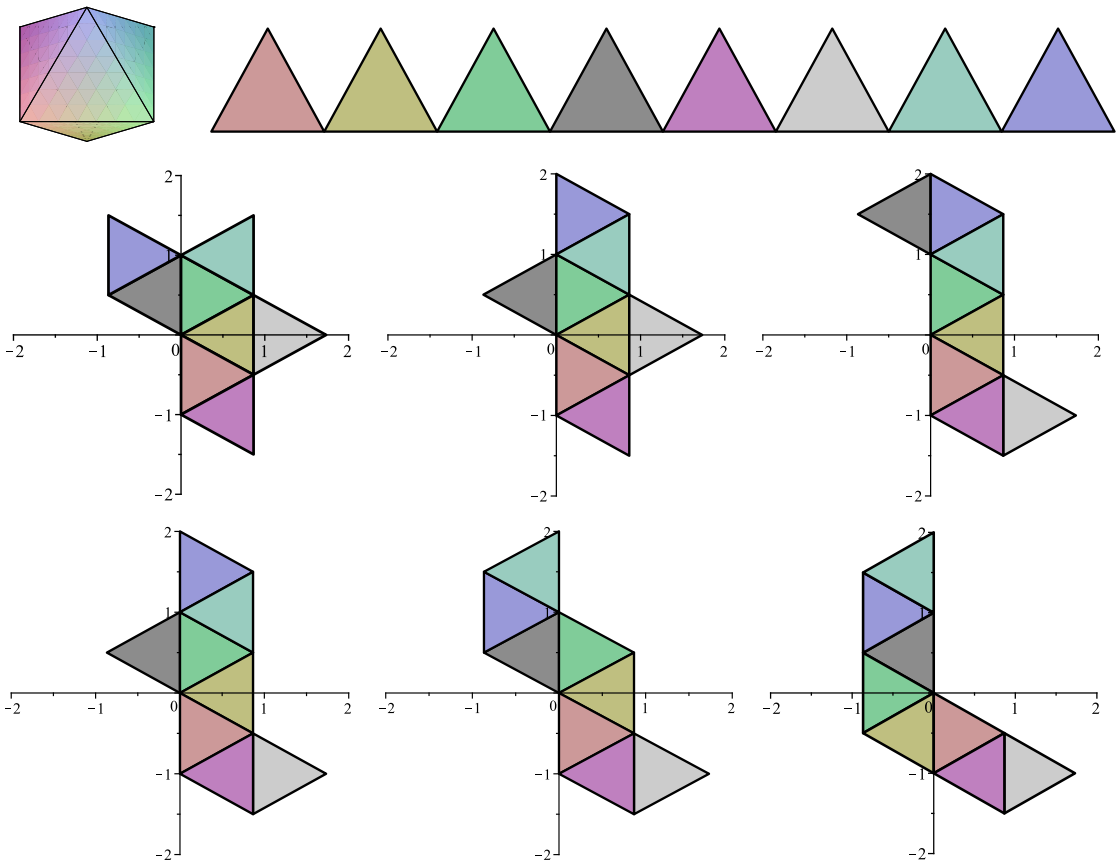
```
    l:=[lu,[lu[1]+s,lu[2]],[lu[1]+s,lu[2]+s],[lu[1],lu[2]+s],lu]:
    p12:=polygon(l,color=farbe,thickness=2):
    plots[display](p12,axes=none,scaling=constrained):
  end proc:

> # Farbpalette
  macro(farbe1=COLOR(RGB,0.80,0.65,0.65)):    # lachs
  macro(farbe2=COLOR(RGB,0.70,0.85,0.70)):    # mint
  macro(farbe3=COLOR(RGB,0.45,0.65,0.60)):    # gruen, dunkel
  macro(farbe4=COLOR(RGB,0.65,0.45,0.70)):    # lila
  macro(farbe5=COLOR(RGB,0.70,0.70,0.45)):    # olive
  macro(farbe6=COLOR(RGB,0.65,0.65,0.85)):    # blau, hell

> display(p_h1([0,0],1,farbe1),p_h1([1,0],1,farbe2),p_h1([2,0],1,farbe3),
          p_h1([3,0],1,farbe4),p_h1([4,0],1,farbe5),p_h1([5,0],1,farbe6),
          scaling =unconstrained);
```



Oktaeder

```
> geom3d[octahedron](f8,point(o,0,0,0),1):
  draw(f8);
> f8:=plottools[octahedron]([0,0,0],1):
      # ra = s/2*sqrt(2) = 1, Radius der Aussenkugel
      # ri = s/6*sqrt(6) = ra/sqrt(3) = 1/sqrt(3), Radius der Innenkugel
      # s = sqrt(2), Kantenlaenge
  display(f8,scaling=constrained);

> # Prozedur fuer Dreieck
  p_o1:=proc(lu,s,drehwinkel,farbe)
    local lh,l,p12,r:    # lokale Parameter
    r:=s/sqrt(3):        # Radius des Umkreises
                         # 2 Ecken des Dreiecks drehen
    lh:=[[0,0],[s*cos(drehwinkel),-s*sin(drehwinkel)],
```

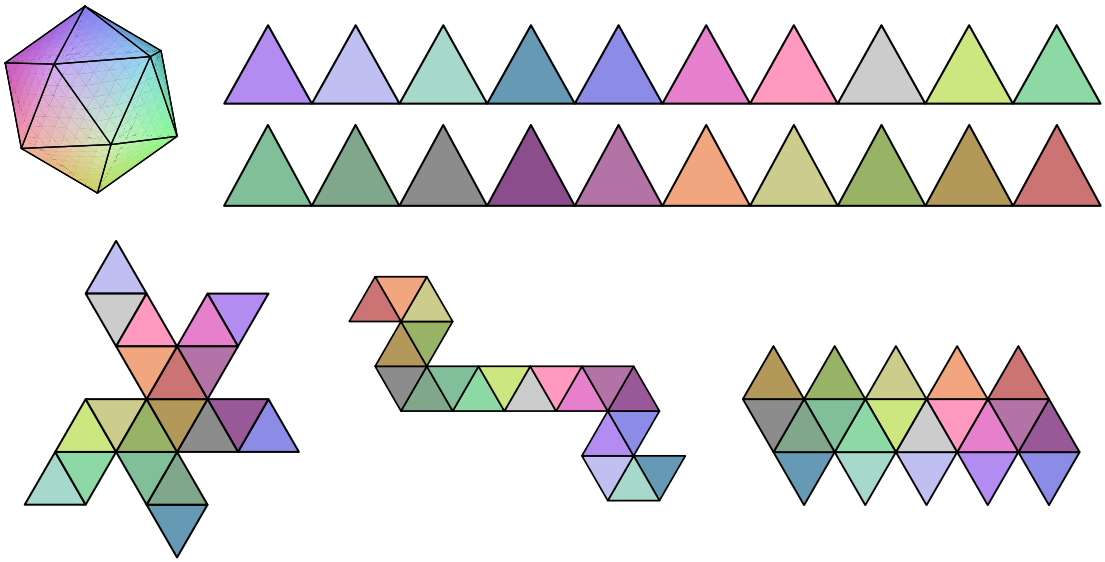
```
                [s/2*(cos(drehwinkel)+sqrt(3)*sin(drehwinkel)),
                 s/2*(-sin(drehwinkel)+sqrt(3)*cos(drehwinkel))]]:
    l:=[lu,[lu[1]+lh[2][1],lu[2]+lh[2][2]],
           [lu[1]+lh[3][1],lu[2]+lh[3][2]],lu]:
    p12:=polygon(l,color=farbe,thickness=2):
    plots[display](p12,axes=none,scaling=constrained):
  end proc:

> # Farbpalette
  macro(farbe1=COLOR(RGB,0.80,0.60,0.60)):    # lachs
  macro(farbe2=COLOR(RGB,0.75,0.75,0.50)):    # olive
  macro(farbe3=COLOR(RGB,0.50,0.80,0.60)):    # gruen
  macro(farbe4=COLOR(RGB,0.55,0.55,0.55)):    # graphit
  macro(farbe5=COLOR(RGB,0.75,0.50,0.75)):    # lila
  macro(farbe6=COLOR(RGB,0.80,0.80,0.80)):    # grau
  macro(farbe7=COLOR(RGB,0.60,0.80,0.75)):    # cyan
  macro(farbe8=COLOR(RGB,0.60,0.60,0.85)):    # blau

> display(p_o1([0,0],1,0,farbe1),p_o1([1,0],1,0,farbe2),
          p_o1([2,0],1,0,farbe3),p_o1([3,0],1,0,farbe4),
          p_o1([4,0],1,0,farbe5),p_o1([5,0],1,0,farbe6),
          p_o1([6,0],1,0,farbe7),p_o1([7,0],1,0,farbe8),
          scaling=unconstrained);
```



Ikosaeder

```
> geom3d[icosahedron](f20,point(o,0,0,0),sqrt(10+2*sqrt(5))/4):
  draw(f20);
> f20:=plottools[icosahedron]([0,0,0],1):
       # 1 = Entfernung von Mitte zu einer Kante
       # ri = 2*sqrt(2)/3=s/12*sqrt(3)*(3+sqrt(5)), Radius der Innenkugel
       # ra = 2/3*sqrt(6)*sqrt(5-2*sqrt(5)) = s/4*sqrt(2*(5+sqrt(5)))
       # ra = ri*sqrt(3)*sqrt(5-2*sqrt(5)), Radius der Aussenkugel
       # s = 2/3*sqrt(6)*(3-sqrt(5)), Kantenlaenge
       # h = s*sqrt(3)/2, Hoehe des Dreiecks
  display(f20,scaling=constrained);

> # Prozedur fuer Dreieck
  p_i1:=proc(lu,s,drehwinkel,farbe)
    local lh,l,p12,r:    # lokale Parameter
    r:=s/sqrt(3):        # Radius des Umkreises
                         # 2 Ecken des Dreiecks drehen
    lh:=[[0,0],[s*cos(drehwinkel),-s*sin(drehwinkel)],
                [s/2*(cos(drehwinkel)+sqrt(3)*sin(drehwinkel)),
                 s/2*(-sin(drehwinkel)+sqrt(3)*cos(drehwinkel))]]:
    l:=[lu,[lu[1]+lh[2][1],lu[2]+lh[2][2]],
           [lu[1]+lh[3][1],lu[2]+lh[3][2]],lu]:
    p12:=polygon(l,color=farbe,thickness=2):
    plots[display](p12,axes=none,scaling=constrained):
  end proc:

> # Farbpalette
  macro(farbe01=COLOR(RGB,0.70,0.55,0.95)):    # lila
  macro(farbe02=COLOR(RGB,0.75,0.75,0.95)):    # alu
  macro(farbe03=COLOR(RGB,0.65,0.85,0.80)):    # cyan
  macro(farbe04=COLOR(RGB,0.40,0.60,0.70)):    # teal
  macro(farbe05=COLOR(RGB,0.55,0.55,0.90)):    # blau
  macro(farbe06=COLOR(RGB,0.90,0.50,0.80)):    # rotlila
  macro(farbe07=COLOR(RGB,1.00,0.60,0.75)):    # altrosa
  macro(farbe08=COLOR(RGB,0.80,0.80,0.80)):    # grau
  macro(farbe09=COLOR(RGB,0.80,0.90,0.50)):    # mint
  macro(farbe10=COLOR(RGB,0.55,0.85,0.65)):    # gruen
  macro(farbe11=COLOR(RGB,0.50,0.75,0.60)):    # gruen, dunkel
  macro(farbe12=COLOR(RGB,0.50,0.65,0.55)):    # blaugruen
  macro(farbe13=COLOR(RGB,0.55,0.55,0.55)):    # graphit
  macro(farbe14=COLOR(RGB,0.55,0.30,0.55)):    # lila, dunkel
  macro(farbe15=COLOR(RGB,0.70,0.45,0.65)):    # altlila
  macro(farbe16=COLOR(RGB,0.95,0.65,0.50)):    # beige
  macro(farbe17=COLOR(RGB,0.80,0.80,0.55)):    # olive, hell
  macro(farbe18=COLOR(RGB,0.60,0.70,0.40)):    # olive
  macro(farbe19=COLOR(RGB,0.70,0.60,0.35)):    # braun
  macro(farbe20=COLOR(RGB,0.80,0.45,0.45)):    # terrakotta

> display(p_i1([0,0],1,0,farbe01), p_i1([1,0],1,0,farbe02),
          p_i1([2,0],1,0,farbe03), p_i1([3,0],1,0,farbe04),
          p_i1([4,0],1,0,farbe05), p_i1([5,0],1,0,farbe06),
          p_i1([6,0],1,0,farbe07), p_i1([7,0],1,0,farbe08),
          p_i1([8,0],1,0,farbe09), p_i1([9,0],1,0,farbe10),
```

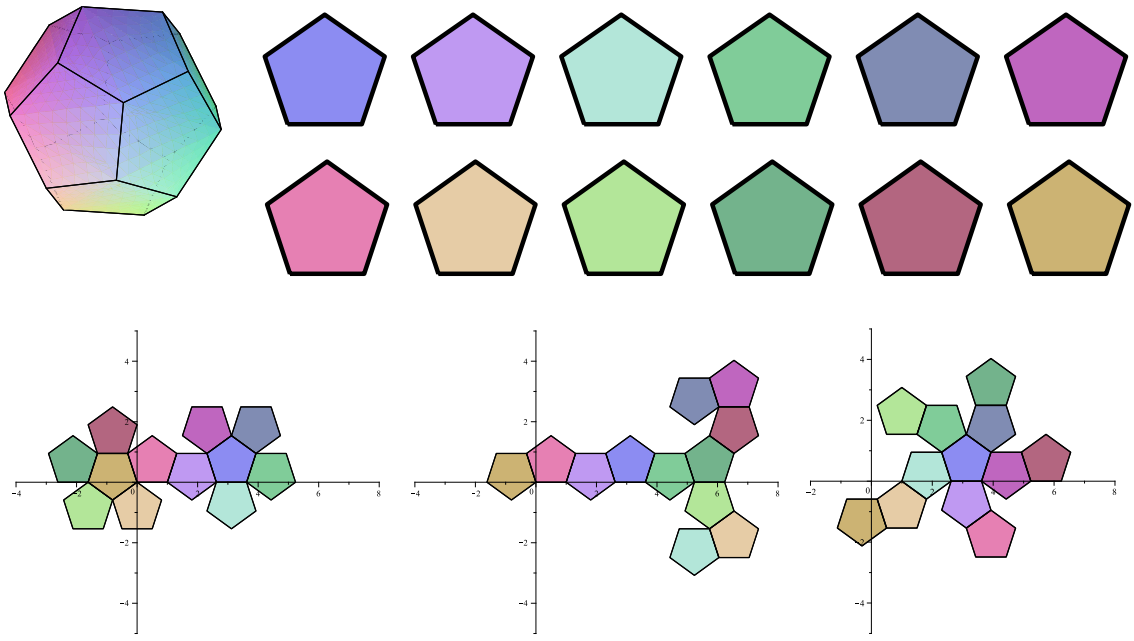
```
        p_i1([10,0],1,0,farbe11),p_i1([11,0],1,0,farbe12),
        p_i1([12,0],1,0,farbe13),p_i1([13,0],1,0,farbe14),
        p_i1([14,0],1,0,farbe15),p_i1([15,0],1,0,farbe16),
        p_i1([16,0],1,0,farbe17),p_i1([17,0],1,0,farbe18),
        p_i1([18,0],1,0,farbe19),p_i1([19,0],1,0,farbe20),
        scaling=unconstrained);
```



Dodekaeder

```
> geom3d[dodecahedron](f12,point(o,0,0,0),sqrt(3)*(1+sqrt(5))/4):
  draw(f12);
> f12:=plottools[dodecahedron]([0,0,0],1):
       # 1 = Entfernung von Mitte zu einer Kante
       # ri = sqrt(3)/2 = s/20*sqrt(10*(25+11*sqrt(5))),
       #      Radius der Innenkugel
       # ra = 3/2*sqrt(5*(3+sqrt(5))/(25+11*sqrt(5)))=s/4*sqrt(3)*(1+sqrt(5))
       # ra = ri*sqrt(3)/sqrt(1+2/sqrt(5)), Radius der Aussenkugel
       # s = sqrt(6/(5+11/sqrt(5))), Kantenlaenge
       # h = s*(sqrt((5+2*sqrt(5))/20)+sqrt((5+sqrt(5))/10)),
       #     Hoehe des Fuenfecks
  plots[display](f12,scaling=constrained,orientation=[60,0]);

> # Prozedur fuer Fuenfeck
  p_d1:=proc(lu,s,drehwinkel,farbe)
    local lh,l,p1,p2,r,h:           # lokale Parameter
    r:=s*2/sqrt(10-2*sqrt(5)):      # Radius des Umkreises
    h:=sqrt(r^2-(s/2)^2):           # Hoehe im gleichschenkeligen Dreieck
                                    # des Fuenfecks
    lh:=[[0,0],[s*cos(drehwinkel),-s*sin(drehwinkel)],
     [ s*(1+cos(2*Pi/5))*cos(drehwinkel)+s*sin(2*Pi/5)*sin(drehwinkel),
      -s*(1+cos(2*Pi/5))*sin(drehwinkel)+s*sin(2*Pi/5)*cos(drehwinkel)],
     [ s/2*cos(drehwinkel)+(h+r)*sin(drehwinkel),
      -s/2*sin(drehwinkel)+(h+r)*cos(drehwinkel)],
     [-s*cos(2*Pi/5)*cos(drehwinkel)+s*sin(2*Pi/5)*sin(drehwinkel),
       s*cos(2*Pi/5)*sin(drehwinkel)+s*sin(2*Pi/5)*cos(drehwinkel)]]:
```

```
    l:=[lu,
        [lu[1]+lh[2][1],lu[2]+lh[2][2]],[lu[1]+lh[3][1],lu[2]+lh[3][2]],
        [lu[1]+lh[4][1],lu[2]+lh[4][2]],[lu[1]+lh[5][1],lu[2]+lh[5][2]],
        lu]:
    p1:=polygon(l,color=farbe):
    p2:=plot(l,color=black,thickness=2):
    plots[display]([p1,p2],axes=none,scaling=constrained):
  end proc:

> # Farbpalette
  macro(farbe01=COLOR(RGB,0.55,0.55,0.95)):    # blau
  macro(farbe02=COLOR(RGB,0.75,0.60,0.95)):    # lila
  macro(farbe03=COLOR(RGB,0.70,0.90,0.85)):    # cyan
  macro(farbe04=COLOR(RGB,0.50,0.80,0.60)):    # gruen
  macro(farbe05=COLOR(RGB,0.50,0.55,0.70)):    # stahl
  macro(farbe06=COLOR(RGB,0.75,0.40,0.75)):    # lila dunkel
  macro(farbe07=COLOR(RGB,0.90,0.50,0.70)):    # rosa
  macro(farbe08=COLOR(RGB,0.90,0.80,0.65)):    # beige
  macro(farbe09=COLOR(RGB,0.70,0.90,0.60)):    # mint
  macro(farbe10=COLOR(RGB,0.45,0.70,0.55)):    # gruen dunkel
  macro(farbe11=COLOR(RGB,0.70,0.40,0.50)):    # lachs dunkel
  macro(farbe12=COLOR(RGB,0.80,0.70,0.45)):    # braun

> display(p_d1([0,0],1,0,farbe01),p_d1([2,0],1,0,farbe02),
          p_d1([4,0],1,0,farbe03),p_d1([6,0],1,0,farbe04),
          p_d1([8,0],1,0,farbe05),p_d1([10,0],1,0,farbe06),
          p_d1([12,0],1,0,farbe07),p_d1([14,0],1,0,farbe08),
          p_d1([16,0],1,0,farbe09),p_d1([18,0],1,0,farbe10),
          p_d1([20,0],1,0,farbe11),p_d1([22,0],1,0,farbe12),
          scaling=unconstrained);
```

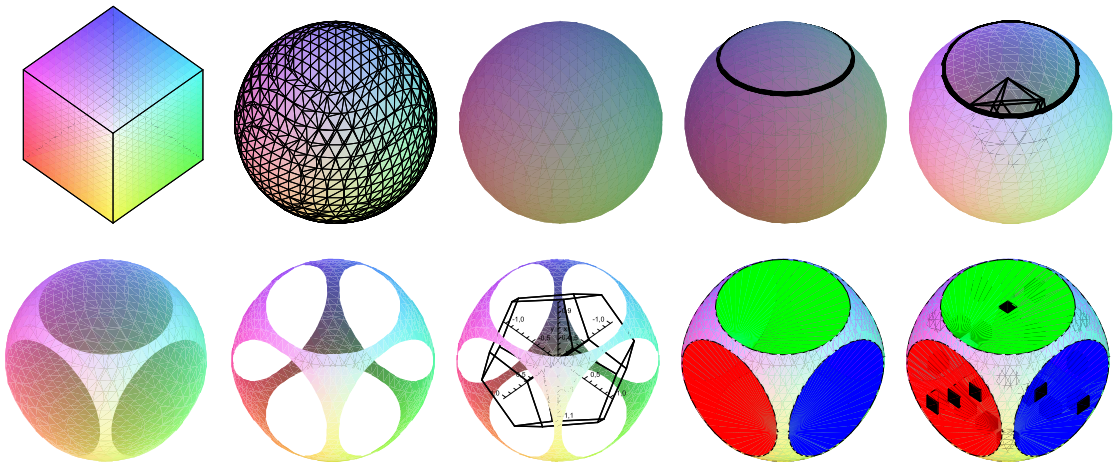


2.6 Runde Sachen

2.6.1 Zahlenwürfel

Verzeichnis: `zahlenwuerfel`
Dateien: zahlenwuerfel1.mw, zeige_mal6.mws
Modelle: Kugel, Tischtennisball, Würfel

Konstruktion des Zahlenwürfels aus der gekappten Kugel



Maple-Anweisungen zum Würfel

```
> # Kugel, Kugel kappen, Kugelloecher mit bunten Kreisen fuellen
> ri:=evalf(2*sqrt(2)/3):
  r:=ri+0.4:
  f202b := implicitplot3d(x^2+y^2+z^2=r^2,x=-1..1,y=-1..1,z=-1..1,
                          style=patchnogrid,grid=[20,20,20]):
  rk:=sqrt(r^2-1^2):
  n:=40:
  h:=2*Pi/n:
  soben:=[seq([rk*cos(h*i),rk*sin(h*i),1],i=0..n)]:
  loben:=polygon(soben,color=green,thickness=1):
  sunten:=[seq([rk*cos(h*i),rk*sin(h*i),-1],i=0..n)]:
  lunten:=polygon(sunten,color=aquamarine,thickness=1):
  srechts:=[seq([rk*cos(h*i),1,rk*sin(h*i)],i=0..n)]:
  lrechts:=polygon(srechts,color=blue,thickness=1):
  slinks:=[seq([rk*cos(h*i),-1,rk*sin(h*i)],i=0..n)]:
  llinks:=polygon(slinks,color=cyan,thickness=1):
  svorne:=[seq([1,rk*cos(h*i),rk*sin(h*i)],i=0..n)]:
  lvorne:=polygon(svorne,color=red,thickness=1):
  shinten:=[seq([-1,rk*cos(h*i),rk*sin(h*i)],i=0..n)]:
  lhinten:=polygon(shinten,color=magenta,thickness=1):

> plots[display](f202b,loben,lunten,lrechts,llinks,lvorne,lhinten,
                 scaling=constrained);
```

```
> # Zahlenwerte 1..6 hinzufuegen
  ho:=0.92: di:=0.4:  sy:=solidbox:
  eins:=pointplot3d([0,0,ho],symbol=sy,symbolsize=40,color=black):
  sechs:=pointplot3d([[di,di,-ho],[di,0,-ho],[di,-di,-ho],[-di,di,-ho],
         [-di,0,-ho],[-di,-di,-ho]],symbol=sy,symbolsize=50,color=black):
  zwei:=pointplot3d([[di,ho,di],[-di,ho,-di]],
                    symbol=sy,symbolsize=50,color=black):
  fuenf:=pointplot3d([[0,-ho,0],[-di,-ho,-di],[-di,-ho,di],[di,-ho,di],
         [di,-ho,-di]],symbol=sy,symbolsize=50,color=black):
  drei:=pointplot3d([[ho,0,0],[ho,di,di],[ho,-di,-di]],
                    symbol=sy,symbolsize=50,color=black):
  vier:=pointplot3d([[-ho,di,di],[-ho,di,-di],[-ho,-di,-di],
        [-ho,-di,di]],symbol=sy,symbolsize=50,color=black):
> plots[display](f202b,loben,lunten,lrechts,llinks,lvorne,lhinten,
                 eins,zwei,drei,vier,fuenf,sechs,scaling=constrained);
```

2.6.2 Ikosaeder und Fußball

Verzeichnisse: `entfalten_von_koerpern`
Dateien: Bastelbogen_Abgest_Ikosaeder1.pdf, Bastelbogen_Fussball.pdf
Modelle: Bastelbogen, Ikosaeder, Fußball

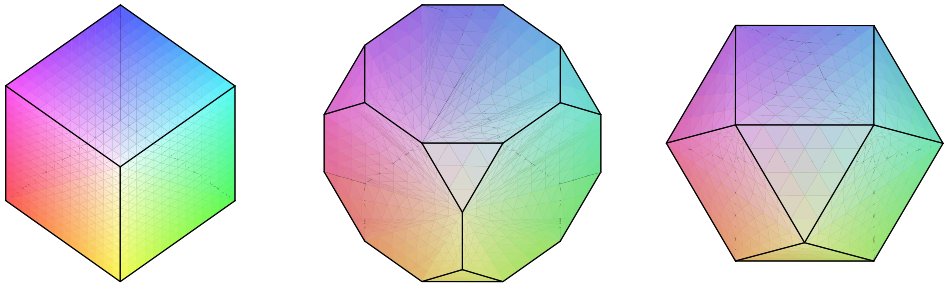
Ikosaeder ⇒ Abgestumpfter Ikosaeder ⇒ Fußball

2.6.3 Polyeder als Fußbälle?

Verzeichnis: `rund_um_den_fussball`
Dateien: fussball_hexa_okta_ikosaeder.mws, fussball_tetraeder.mws, zeige_mal6.mws
Modelle: Fußball, Hexaeder, Oktaeder, Ikosaeder, Tetraeder

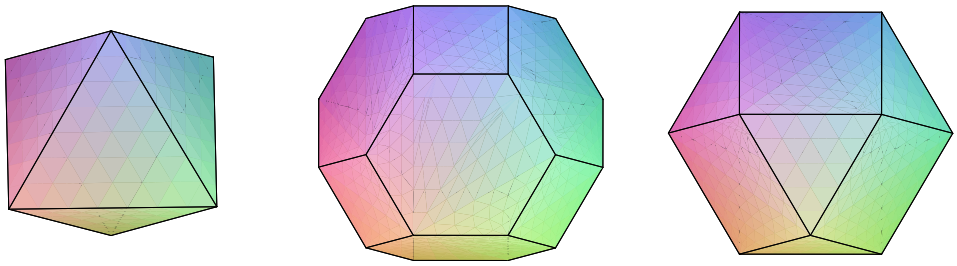
Kann man sich Polyeder bzw. ihre an den Ecken abgestumpften Varianten als Fußbälle vorstellen? Machen wir einen grafischen Versuch mit Maple-Berechnungen.

Hexaeder, abgestumpfter Hexaeder und kubischer Oktaeder



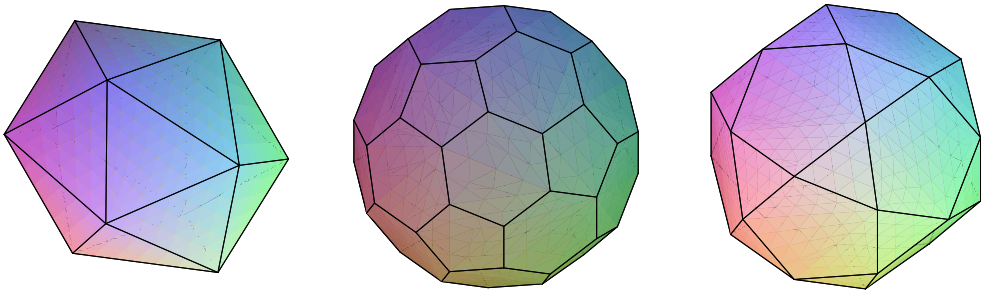
```
> with(plots): with(plottools):   # bzw.
  with(geometry): with(geom3d):

> # Hexaeder
  f6a:=hexahedron([0,0,0],1):       # 1 = Radius der Innenkugel
  plots[display](f6a);
> # Abgestumpfter Hexaeder:
  # 8 gleichseitige Dreiecke, 6 Achtecke (regelmaessig oder nicht)
  TruncatedHexahedron(t13,point(o,0,0,0),1):
  draw(t13);
> # Kubischer Oktaeder:  8 gleichseitige Dreiecke, 6 Quadrate
  cuboctahedron(t12,point(o,0,0,0),1):
  draw(t12);
```

Oktaeder, abgestumpfter Oktaeder und kubischer Oktaeder



```
> # Oktaeder
  f8a:=octahedron([0,0,0],1):       # 1 = Radius der Aussenkugel
  plots[display](f8a,orientation=[40,75]);
> # Abgestumpfter Oktaeder
  # 6 Quadrate, 8 Sechsecke (regelmaessig oder nicht regelmaessig
  TruncatedOctahedron(t14,point(o,0,0,0),1):
  draw(t14);
> # Kubischer Oktaeder
  # Kantenlaenge = 1
  QuasiRegularPolyhedron(t11,[[3],[4]],point(o,0,0,0),1):
  draw(t11);
  # spezielle Kommandos dafuer
  cuboctahedron(t12,point(o,0,0,0),1):
    # geom3d[cuboctahedron](t2,point(o,0,0,0),1):
  draw(t12);
```

Ikosaeder und zwei abgestumpfte Ikosaeder

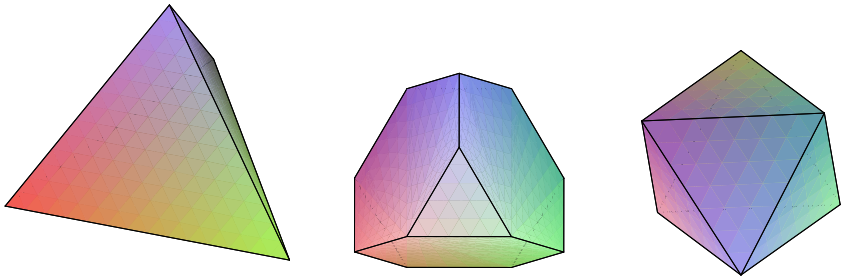


```
> # Ikosaeder
  f20:=icosahedron([0,0,0],1):  # 1 = Entfernung von Mitte zur Kante
  plots[display](f20,orientation=[30,-10]);

> # Abgestumpfter Ikoseder: geringe Abstumpfung
  # 12 Fuenfecke
  # 20 Sechsecke (regelmaessig oder nicht r. -> 20 regelm. Sechsecke)
  # Zaehlung der Sechsecke:
  # jedes der 12 Fuenfecke hat 5 benachbarte Sechsecke
  #   -> 12*5=60 Sechsecke,
  # jedes Sechseck gehoert als Nachbar zu 3 Fuenfecken,
  # wurde also dreimal gezaehlt
  #   -> Anzahl = 60/3=20
  TruncatedIcosahedron(t23,point(o,0,0,0),1):
  draw(t23);

  # Abgestumpfter Ikoseder: groessere Abstumpfung
  #                         Ecken abstumpfen bis zur Seitenmitte
  # 12 Fuenfecke, 20 gleichseitige Dreiecke
  QuasiRegularPolyhedron(t21,[[3],[5]],point(o,0,0,0),1):
  draw(t21);
  # spezielle Kommandos dafuer
  icosidodecahedron(t22,point(o,0,0,0),1):
    # geom3d[icosidodecahedron](t22,point(o,0,0,0),1):
  draw(t22);
```

Als Kind und Jugendlicher habe ich jahrelang mit einem Lederball gespielt (siehe Abbildung). Der Ikosaeder und seine abgestumpfte Version haben sich als Muster für die Formgestaltung des Fußballs natürlich angeboten. Aus der Dreieckseite entstehen die neuen Oberflächenelemente Sechseck und Fünfeck, die wiederum konvex sind.

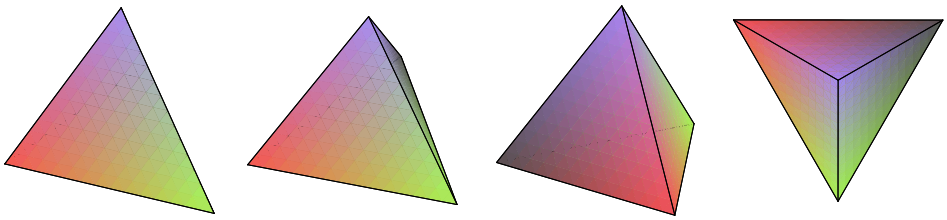
Dass der Hexaeder Ausgangspunkt für den Fußball sein kann, ist nicht so leicht vorstellbar. Aber die Verwendung der Paneelenstruktur macht es möglich, wobei mit den Turbinen- und Propeller-Elementen die Konvexität verloren geht.
Aber eigentlich lag es nun nahe, auch einen Mittelweg zu finden.
Dazu empfiehlt sich wider Erwarten der Tetraeder, obwohl er 4 spitze Ecken besitzt. Die Abstumpfung der Ecken bis zur Kantenmitte liefert der Oktaeder. Eine etwas geringere Abstumpfung der Ecken führen zu gleichseitigen Dreiecken, die abgerundet werden können. Es bleiben auf den Seiten regelmäßige bzw. unregelmäßige Sechsecke übrig, die aber in 4 Teile zerlegt als Paneele geformt werden können (Achtung! Es werden aber 8 Formen von Paneelen verwendet).
Gibt es vielleicht noch weitere Lösungsvarianten für den Fußball? Ich denke schon. Aber man muss dabei auch die ballistischen Eigenschaften sowie produktionstechnischen Möglichkeiten berücksichtigen.

Tetraeder und zwei abgestumpfte Tetraeder (letzterer ist Oktaeder)
Die Betrachtungen werden etwas aufführlicher gemacht.



```
> # Tetraeder
  f4:=tetrahedron([0,0,0],1):  # 1 = Entfernung von Mitte zur Kante
  plots[display](f4,scaling=constrained);

> # Abgestumpfter Tetraeder: geringe Abstumpfung
  # 4 Dreiecke
  # 4 Sechsecke (regelmaessig, Ecken abstumpfen zu 1/3 der Kantenlaenge
  #              oder nicht regelmaessig Sechsecke)
  TruncatedTetrahedron(t6,point(o,0,0,0),1): # 1=Radius der Aussenkugel
  draw(t6);

> # Abgestumpfter Ikoseder: groessere Abstumpfung
  #                         Ecken abstumpfen bis zur Kantenmitte
  # 8 gleichseitige Dreiecke -> Oktaeder
  t8:=octahedron([0,0,0],1):   # 1 = Radius der Aussenkugel
  display(t8,orientation=[40,-60]);
```

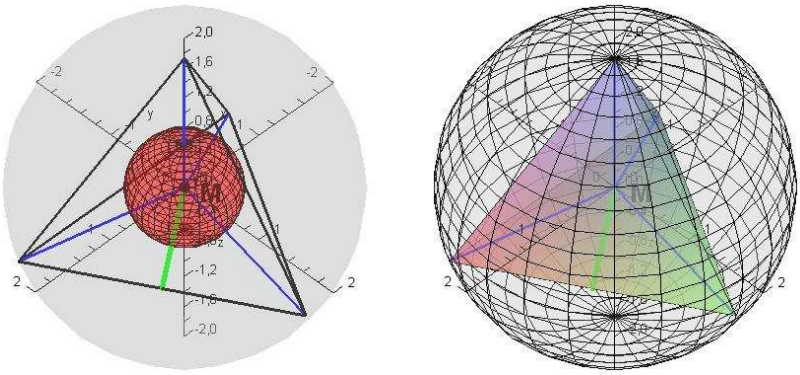
Tetraeder mit 4 Ansichten



```
> # Tetraeder mit 4 Perspektiven
  f4:=tetrahedron([0,0,0],1):
      # 1 = Entfernung von Mitte zu Kante
      # ra = sqrt(3), Radius der Aussenkugel
      # s = 2*sqrt(2), Kantenlaenge
      # h = s*sqrt(3)/2, Hoehe des Dreiecks

> display(f4);                      # scaling=unconstrained
  display(f4,scaling=constrained);
  display(f4,scaling=constrained,orientation=[-55,65]);
  display(f4,scaling=constrained,orientation=[90,0]);
> # Groessen des Tetraeders bei 1 = Entfernung von Mitte zu Kante
  # Mitte/Schwerpunkt liegt auf der Hoehe im Abstand von H/4
  # von der Grundflaeche
```

```
ra:=sqrt(3);
evalf(ra);          # Radius der Aussenkugel, ra=s/4*sqrt(6)
ri:=1/sqrt(3);
evalf(ri);          # Radius der Innenkugel,  ri=s/12*sqrt(6)
s:=2*sqrt(2);
evalf(s);           # Kantenlaenge, s
e:=1;               # Entfernung von Mitte zu Kante, e=s/4*sqrt(2)
h:=sqrt(6);
evalf(h);           # Hoehe des gleichseit. Dreiecks, h=s/2*sqrt(3)
H:=4/sqrt(3);
evalf(H);           # Hoehe des Tetraeders, H=s/3*sqrt(6)
V:=s^3*sqrt(2)/12;
evalf(V);           # Volumen, V=s^3*sqrt(2)/12
A:=s^2*sqrt(3);
evalf(A);           # Oberflaeche, A=s^2*sqrt(3)
```

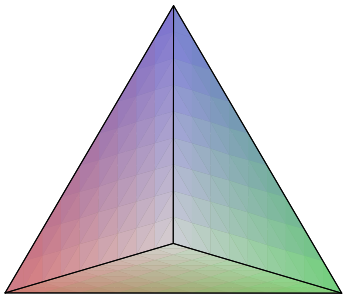
Tetraeder mit Innen- und Außenkugel (In- und Umkugel)



```
> # Bild links
  M:=[0,0,0]:
  t41:=display(tetrahedron(M,1),style=wireframe,thickness=2,
               color=black):
  t42:=sphere(M,ri,color=red,transparency=0.7):
  t43:=sphere(M,ra,color=gray,transparency=0.7,style=patchnogrid):
  t44:=pointplot3d(M,symbol=solidcircle,symbolsize=16,color=black):
  t45:=pointplot3d([M+[0,0,3*H/4],M,M+[-s/2,-h/3,-H/4],M,
                    M+[s/2,-h/3,-H/4],M,M+[0,2*h/3,-H/4]],
                    style=line,color=blue,thickness=2):
  t46:=pointplot3d([M+[s/4,h/6,-H/4],M],style=line,color=green,
                    thickness=3):
  t47:=textplot3d([-0.1,0.2,0,` M`],font=[HELVETICA,BOLD,20],
                  color=black):
  display(t41,t42,t43,t44,t45,t46,t47,view=[-2..2,-2..2,-2..2],
          axes=normal,scaling=constrained,labels=[` x`,` y`,` z`]);

> # Bild rechts
  t48:=display(tetrahedron([0,0,0],1),transparency=0.4):
  t49:=sphere(M,ra,color=gray,transparency=0.8):
  display(t45,t46,t47,t48,t49,view=[-2..2,-2..2,-2..2],
          axes=normal,scaling=constrained,labels=[` x`,` y`,` z`]);
```

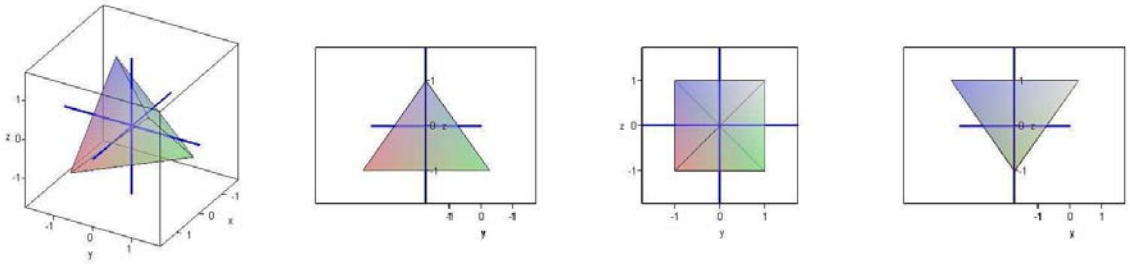
Mit dem Befehl `tetrahedron` aus dem Paket `geom3d` kann man ebenfalls den Tetraeder zeichnen. Sein Parameterkonzept unterscheidet sich jedoch von dem aus dem Paket `plottools`. Dazu besteht die Möglichkeit, die wichtigsten Parameter zum Polyeder selber zu erhalten.

```
> with(geom3d):
  tetrahedron(t53,point(o,0,0,0),1):  # Mitte = M = [0,0,0]
                                      # Radius der Aussenkugel = 1
  draw(t53);
```



```
> tetrahedron(t54,point(o,0,0,0),sqrt(3)):  # Mitte = M = [0,0,0]
                                  # Radius der Aussenkugel = sqrt(3)

  pl1:=draw(t54,axes=boxed,labels=[‘ x‘,‘ y‘,‘ z‘],transparency=0.5,
            orientation=[30,60],tickmarks=[3,3,3]):
  pl2:=pointplot3d([[-2,0,0],[2,0,0]],style=line,color=blue,thickness=2):
  pl3:=pointplot3d([[0,-2,0],[0,2,0]],style=line,color=blue,thickness=2):
  pl4:=pointplot3d([[0,0,-2],[0,0,2]],style=line,color=blue,thickness=2):

> display(pl1,pl2,pl3,pl4);
```

Der mit diesem Kommando erzeugte Tetraeder hat erst einmal nicht den Anblick einer Pyramide, wo man eine Grundfläche in der horizontalen Ebene erwartet. Er liegt schief im Raum und Draufsichten von oben bzw. von der Seite zeigen einen dreieckigen bzw. quadratischen Umriss. Die Diagonalen des Quadrats sind die gegenüber und zueinander schief (rechtwinkelig) liegenden Kanten des Tetraeders. Mit dem Außenkugelradius $r_a = \sqrt{3}$ haben die 4 Tetraederecken die Koordinatenform $(\pm 1, \pm 1, \pm 1)$.



Wollen wir die Daten des Tetraeders `f4` mit denen von `t54` vergleichen, wo die Entfernung von Mitte zur Kante gleich 1 ist, schreiben wir die folgenden Anweisungen.

```
> # Kontrolle und Vergleich der Parameter
  radius(t54);          # Radius der Aussenkugel
  form(t54);            # Typ des Polyeders
  schlafli(t54);
  center(t54);          # Schwerpunkt, Mitte
  M:=[0,0,0];

> ra:=radius(t54);      # Radius der Aussenkugel, ra=s/4*sqrt(6)
  ri:=InRadius(t54);    # Radius der Innenkugel,  ri=s/12*sqrt(6)

  s:=sides(t54);        # Kantenlaenge, s
  e:=MidRadius(t54);    # Entfernung von Mitte zu Kante, e=s/4*sqrt(2)
  V:=volume(t54);       # Volumen, V=s^3*sqrt(2)/12
  A:=area(t54);         # Oberflaeche, A=s^2*sqrt(3)

  h:=s/2*sqrt(3);       # Hoehe des gleichseit. Dreiecks, h=s/2*sqrt(3)
  H:=s/3*sqrt(6);       # Hoehe des Tetraeders, H=s/3*sqrt(6)

  Seiten:=faces(t54);   # 4 Seiten zu je 3 Ecken
  Ecken:=vertices(t54); # 4 Tetraederecken
```

$$tetrahedron3d$$

$$[3,3]$$

$$o$$

$$M := [0,0,0]$$

$$ra := \sqrt{3}$$

$$ri := \frac{1}{6}\sqrt{6}\sqrt{2}$$

$$s := 2\sqrt{2}$$

$$e := 1$$

$$V := \frac{8}{3}$$

$$A := 8\sqrt{3}$$

$$h := \sqrt{2}\sqrt{3}$$

$$H := \frac{2}{3}\sqrt{6}\sqrt{2}$$

$$Seiten := [[[1,1,1],[1,-1,-1],[-1,1,-1]],\ [[1,1,1],[-1,-1,1],[1,-1,-1]],\ [[1,1,1],[-1,1,-1],[-1,-1,1]],\ [[1,-1,-1],[-1,-1,1],[-1,1,-1]]]$$

$$Ecken := [[1,1,1],[1,-1,-1],[-1,1,-1],[-1,-1,1]]$$

Für die Anschauung ist die Lage des Tetraeders `f4` günstiger.
Er hat den Außenradius $r_a = \sqrt{3}$ und die Entfernung 1 von Mitte zu Kante.
Mit seiner Mitte im Koordinatenursprung kann er ohne Probleme proportional gestreckt werden. Aus den folgenden Anweisungen entnimmt man die räumlichen Koordinaten seiner 4 Eckpunkte sowie eines Kantenmittelpunktes.

```
M:=[0,0,0]:
t45:=pointplot3d([M+[0,0,3*H/4],M,M+[-s/2,-h/3,-H/4],M,
                  M+[s/2,-h/3,-H/4],M,M+[0,2*h/3,-H/4]],
                  style=line,color=blue,thickness=2):
t46:=pointplot3d([M+[s/4,h/6,-H/4],M],style=line,color=green,
                  thickness=3):
```

Die Streckung q der Pyramide beziehen wir in die Formel für die Kantenlänge s ein, so dass sie auch in alle anderen abgeleiteten Größen eingeht.

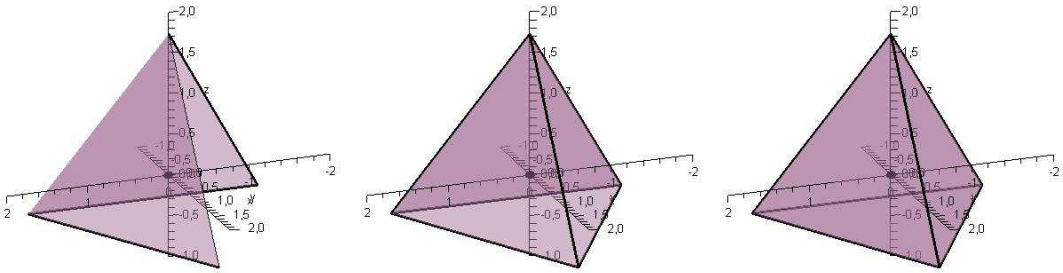
```
> # vierseitige Pyramide mit Entfernung von Mitte zu Kante, e=1
  q:=1;                    # Streckung, q=31/(2*sqrt(2)) -> s=31

  s:=q*2*sqrt(2);          # Kantenlaenge
  ra:=s/4*sqrt(6);         # Radius der Aussenkugel
  ri:=s/12*sqrt(6);        # Radius der Innenkugel
  e:=s/4*sqrt(2);          # Entfernung von Mitte zu Kante
  h:=s/2*sqrt(3);          # Hoehe des gleichseitigen Dreiecks
  H:=s/3*sqrt(6);          # Hoehe des Tetraeders

> # Punkte und Seiten
  M:=[0,0,0];              # Mitte/Schwerpunkt des Tetraeders
                           # auf H/4 der Tetraederhoehe
  SM:=M+[s/4,h/6,-H/4];    # Kantenmitte

                           # 4 Tetraederecken
  Eck:=[M+[0,0,3*H/4],M+[-s/2,-h/3,-H/4],
        M+[s/2,-h/3,-H/4],M+[0,2*h/3,-H/4]];

                           # 4 Seiten zu je 3 Ecken
  Sei:=[[Eck[1],Eck[2],Eck[3]],[Eck[1],Eck[3],Eck[4]],
        [Eck[1],Eck[4],Eck[1]],[Eck[2],Eck[3],Eck[4]]];
```

Zunächst machen wir einen Test zum Zeichnen der Pyramide.

```
> # Pyramide, Grundflaeche und 1 Seite offen
  pp1:=polygonplot3d(Eck,color=farbe01,transparency=0.5,
              orientation=[70,70],axes=normal,labels=[‘x‘,‘y‘,‘z‘]):
  pp2:=pointplot3d(Eck,style=line,color=black,thickness=2):
  pp3:=pointplot3d(M,symbol=solidcircle,symbolsize=16,color=black):
  display(pp1,pp2,pp3,scaling=constrained,
                      view=[-2*q..2*q,-q..2*q,-q..2*q]);

> # Pyramide, Grundflaeche offen
  pp4:=polygonplot3d([op(Eck),Eck[2]],color=farbe01,transparency=0.5,
              orientation=[70,70],axes=normal,labels=[‘x‘,‘y‘,‘z‘]):
```

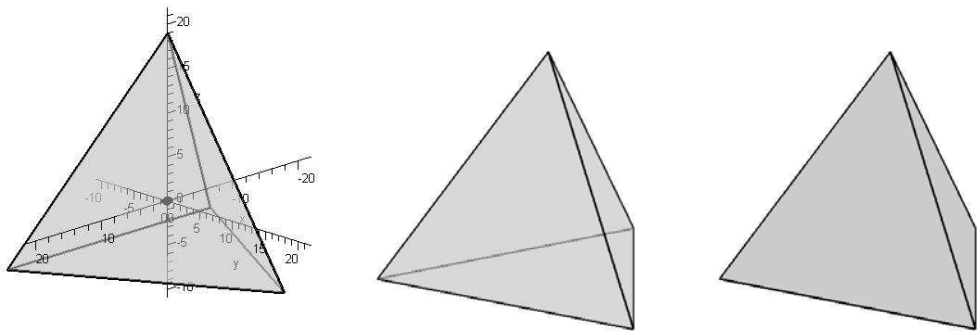
```
pp5:=pointplot3d([op(2..4,Eck),Eck[2],Eck[1],Eck[3],Eck[4],Eck[1]],
            style=line,color=black,thickness=2):
display(pp3,pp4,pp5,scaling=constrained,
                    view=[-2*q..2*q,-q..2*q,-q..2*q]);
```

```
> # Pyramide, alle Seiten geschlossen
pp6:=polygonplot3d(Sei[4],color=farbe01,transparency=0.5):
# pp6:=polygonplot3d([op(2..4,Eck)],color=farbe01,transparency=0.5):
display(pp3,pp4,pp5,pp6,scaling=constrained,
                        view=[-2*q..2*q,-q..2*q,-q..2*q]);
```



```
> pp3:=pointplot3d(M,symbol=solidcircle,symbolsize=16,color=black):
pp4:=polygonplot3d([op(Eck),Eck[2]],color=farbe18,transparency=0.5,
            orientation=[60,70],axes=normal,labels=[‘x‘,‘y‘,‘z‘]):
pp4a:=polygonplot3d([op(Eck),Eck[2]],color=farbe18,transparency=0,
            orientation=[60,70],axes=normal,labels=[‘x‘,‘y‘,‘z‘]):
pp5:=pointplot3d([op(2..4,Eck),Eck[2],Eck[1],Eck[3],Eck[4],Eck[1]],
                style=line,color=black,thickness=2):
pp6:=polygonplot3d(Sei[4],color=farbe18,transparency=0.5):
```

```
> display(pp3,pp4,pp5,pp6,scaling=constrained,
                        view=[-2*q..2*q,-q..2*q,-q..2*q]);
display(pp4,pp5,pp6,scaling=constrained,axes=none);
display(pp4a,pp5,pp6,scaling=constrained,axes=none);
```

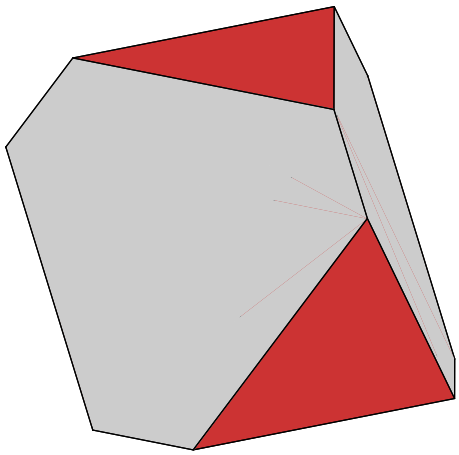


Für den Tetraeder mit einer Kantenlänge $s = 31$ benötigen wir den Streckungskoeffizienten $q = \frac{31}{2\sqrt{2}}$.
Die “geringere“ Abstumpfung der Tetraederecken mit dem Kantenabschnitt $s\frac{13}{31} \approx s\frac{2}{5}$ führt zum Ausgangskörper für den Fußball.

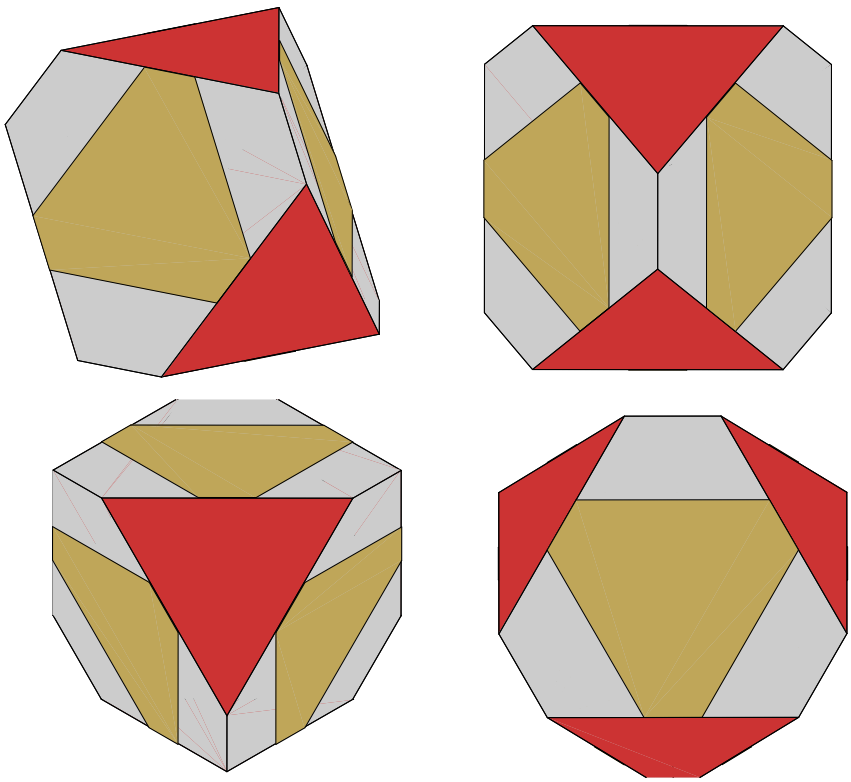
Der abgestumpfte Tetraeder besitzt auf seiner Oberfläche 4 gleichseitige Dreiecke mit der Seitenlänge 13 sowie 4 (unregelmaessige) Sechsecke mit den Seitenlängen 13 und 5. Jedes Sechseck unterteilen wir nochmal in ein Sechseck (Seitenlängen 10 und 3) und 3 Trapeze (Grundseite = 10, andere Seiten = 5). Wir sehen, die Ganzzahligkeit der Größen ist bedingt eben durch die Wahl von $s = 31$.

Es folgen nun die Schritte der Abstumpfung des Polyeders und der weiteren Unterteilung seiner Seiten.

```
> # Tetraederecken abstumpfen --> Dreiecke an den Ecken
  # oben, rechts, links, vorne
  t1:=13/31:
  Do:=[1/31*(18*Eck[1]+13*Eck[2]),
       1/31*(18*Eck[1]+13*Eck[3]),
       1/31*(18*Eck[1]+13*Eck[4])];
  Dr:=[1/31*(18*Eck[2]+13*Eck[1]),
       1/31*(18*Eck[2]+13*Eck[3]),
       1/31*(18*Eck[2]+13*Eck[4])];
  Dl:=[1/31*(18*Eck[3]+13*Eck[1]),
       1/31*(18*Eck[3]+13*Eck[2]),
       1/31*(18*Eck[3]+13*Eck[4])];
  Dv:=[1/31*(18*Eck[4]+13*Eck[1]),
       1/31*(18*Eck[4]+13*Eck[2]),
       1/31*(18*Eck[4]+13*Eck[3])];

> Verbleibende unregelmaessige Sechsecke
  # unten, rechts, hinten, links
  Su:=[Dr[2],Dr[3],Dv[2],Dv[3],Dl[3],Dl[2]];
  Sr:=[Do[3],Do[1],Dr[1],Dr[3],Dv[2],Dv[1]];
  Sh:=[Dl[2],Dl[1],Do[2],Do[1],Dr[1],Dr[2]];
  Sl:=[Dv[1],Dv[3],Dl[3],Dl[1],Do[2],Do[3]];

> # Abgestumpfter Tetraeder
  pl1:=polygonplot3d([Do,Dr,Dl,Dv],color=farbe31,transparency=0.0):
  pl2:=polygonplot3d([Su,Sr,Sh,Sl],color=farbe18,transparency=0.0):
  display(pl1,pl2,scaling=constrained,orientation=[60,70]);
```



```
> # Innere Sechsecke
  # unten, rechts, hinten, links
  t2:=5/13:
  # Korrekturen wegen Hervorheben der inneren Sechsecke
  U:=[0,0,-1]*0.03: U:=[U,U,U,U,U,U]:
  R:=[-1,1,0]*0.03: R:=[R,R,R,R,R,R]:
  L:=[1,1,0]*0.03:  L:=[L,L,L,L,L,L]:
  H:=[0,-1,0]*0.03: H:=[H,H,H,H,H,H]:

  Siu:=[1/13*(8*Dr[2]+5*Dr[3]),1/13*(5*Dr[2]+8*Dr[3]),
        1/13*(8*Dv[2]+5*Dv[3]),1/13*(5*Dv[2]+8*Dv[3]),
        1/13*(8*Dl[3]+5*Dl[2]),1/13*(5*Dl[3]+8*Dl[2])]+U;
  Sir:=[1/13*(8*Do[3]+5*Do[1]),1/13*(5*Do[3]+8*Do[1]),
        1/13*(8*Dr[1]+5*Dr[3]),1/13*(5*Dr[1]+8*Dr[3]),
        1/13*(8*Dv[2]+5*Dv[1]),1/13*(5*Dv[2]+8*Dv[1])]+R;
  Sih:=[1/13*(8*Dl[2]+5*Dl[1]),1/13*(5*Dl[2]+8*Dl[1]),
        1/13*(8*Do[2]+5*Do[1]),1/13*(5*Do[2]+8*Do[1]),
        1/13*(8*Dr[1]+5*Dr[2]),1/13*(5*Dr[1]+8*Dr[2])]+H;
  Sil:=[1/13*(8*Dv[1]+5*Dv[3]),1/13*(5*Dv[1]+8*Dv[3]),
        1/13*(8*Dl[3]+5*Dl[1]),1/13*(5*Dl[3]+8*Dl[1]),
        1/13*(8*Do[2]+5*Do[3]),1/13*(5*Do[2]+8*Do[3])]+L;

> pl3:=polygonplot3d([Siu,Sir,Sih,Sil],color=farbe25,transparency=0.0):
  display(pl1,pl2,pl3,scaling=constrained,orientation=[60,70]);
```

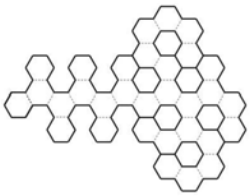
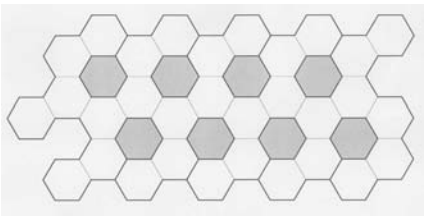


Der abgestumpfte Tetraeder in verschiedenen Perspektiven.

2.6.4 Rund um den Fußball

Verzeichnis: `rund_um_den_fussball`
Dateien: zeige_mal5.mws, fussballplatz.mws, Bastelbogen_Fussball1,3,4.pdf
Modelle: Fußball, Bastelbogen, Ikosaeder, Würfel

Bastelbögen zum Fußball

Das Interessante an diesen Bastelbögen ist, dass sie keine Fünfecke enthalten, sondern nur Sechsecke, obwohl jeder Körper, der nur aus diesen Teilen zusammengesetzt ist, davon 12 Stück braucht. Man schneide entlang der dicken Linien und falte entlang der dünnen Linien. Wenn man die Figur zusammenfaltet, ergibt sich ein Fußball, bei dem die Fünfeckseiten Löcher sind.

Fußball

- $68 - 70\,cm$ Umfang, $410 - 450\,g$ Gewicht, Abweichung von Kugel ca 0.1%
- Spezielle Möglichkeiten
  Ikosidodekaeder: 12 Fünfecke, 20 Dreiecke
  Polyeder (Kämmerling/Jansen 1993): 12 Fünfecke, 60 unregelmäßige Sechsecke
- Bekannte Formen
  Abgestumpfter Ikosaeder (klassisches herkömmliches Modell)
  $\Rightarrow$ **Wabenstruktur (12 Fünfecke, 20 Sechsecke)**


⇒

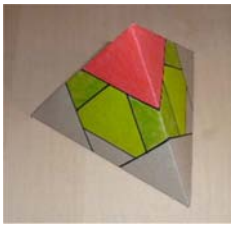
Kubischer Oktaeder (abgestumpfter Hexaeder): 6 Quadrate, 8 Dreiecke
$\Rightarrow$ **Paneelstruktur**
14 Paneele (6 Turbinen, 8 Propeller)


⇒

⇒


Modell: *Adidas Teamgeist 2006*

Abgestumpfter Tetraeder: 4 Dreiecke, 4 Sechsecke, 12 Trapeze
⇒ **Sichtbare Paneelstruktur**
20 Paneele (4 Scheiben als abgerundete Dreiecke, 4 Propeller, 12 Schaufeln)
⇒ **Unsichtbare Paneelstruktur mit 8 Paneelformen**


⇒

⇒


Damit haben wir den Spielball *Jabulani* zur Fußball-WM 2010 in Südafrika.
Es ist das Modell *Adidas Matchball Jabulani WM 2010 Südafrika*.
Die Farbe ist weiß-schwarz, die Größe: EU 5 - UK 5 - US 5.
Die wasserdichte Oberfläche bilden Paneele, wobei 8 verschiedene Paneelformen miteinander verklebt sind und keine Nähte gebraucht werden (Thermal Bonded Technologie). Dazu gibt es eine Grip'n Groove Oberfläche für eine stabilere Flugbahn, eine Latexblase für beste Rücksprungeigenschaften u.a..
Die künstlich aufgedruckten Nähte (Nahtimitationen) lassen eigentlich nur 4 Paneelformen vermuten.




Fußballplatz, Spiel und Torschuss


Im obigen Bild haben wir auf dem Spielfeld `feld` eine **1. Spielerbewegung**:
Der Spieler befindet sich im "Torraumstreifen" und läuft mit Ball ab Mittellinie parallel zur Seitenlinie aufs Tor zu. Bei Annäherung wird der Einschusswinkel größer.

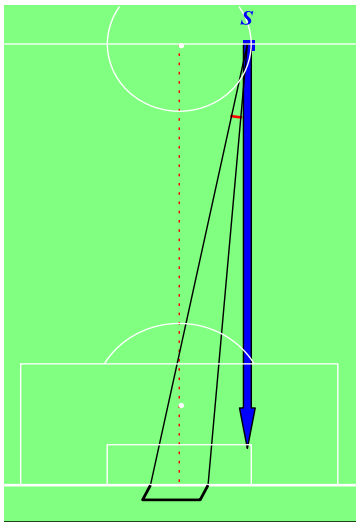
Von allgemeinem Interesse sind Aufstellungen und Bewegungsabläufe von Fußballspielern, Torschusssituationen und günstige Einschussmöglichkeiten.

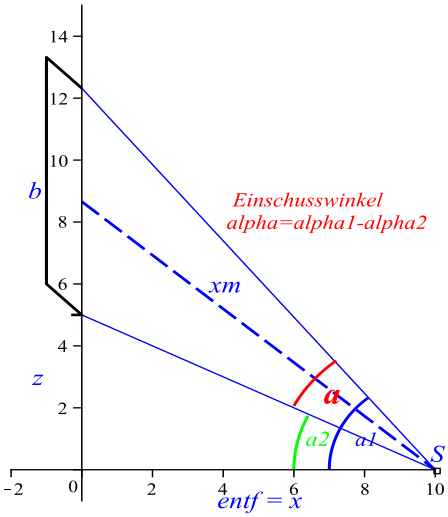
2. Spielerbewegung

Die Praxis. *Am 6.10.2001 spielte die deutsche Fußballnationalmannschaft gegen Finnland. Das Spiel endete torlos 0:0. Die deutsche Boulevard-Presse ging hart mit den Kickern vom damaligen Teram-Chef Rudi Völler ins Gericht. Man wertete das Remis wie eine Niederlage. Die "Dresdner Morgenpost" kritisierte vor allem Stürmer Oliver B., der aus 8 Metern (!) das Tor nicht traf ...*

Warum steht in der Morgenpost hinter der Angabe "8 Meter" das Ausrufezeichen? Sei es vielleicht weniger verwerflich, aus 6 m das Tor zu verfehlen oder aus einer Entfernung von 10 m?

Der Spieler läuft mit Ball ab Mittellinie parallel zur Seitenlinie im Abstand von 5 m vom Tor (Torpfosten) aufs Tor zu. Bei Annäherung an das Tor verändert sich der Einschusswinkel. Gibt es einen größten Winkel?

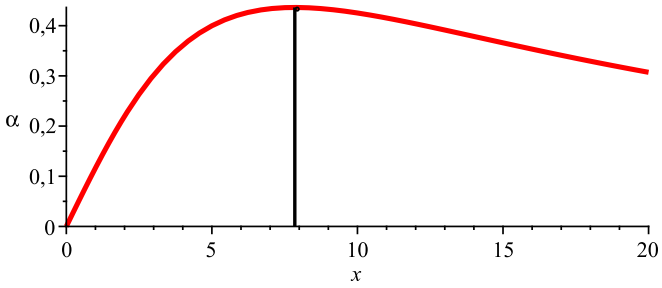





```
> m:=11;      # Elfmeterpunkt
  b:=7.32;    # Torbreite
  b2:=b/2;
  h:=b/3;     # Torhoehe

> qr1:=textplot([[xs,3,'S']],font=[TIMES,ROMAN,8],color=blue):
  qr2:=plot([[-b2,-60],[xs,ys],[b2,-60]],color=black):
  qr3:=plottools[arrow]([xs,ys],[xs,-55],1,2,0.1,color=blue):
  qr4:=pointplot([xs,ys],symbol=box,symbolsize=15,color=blue):
  qr5:=plot([[0,0],[0,-60]],color=red,linestyle=2):
  qr6:=plottools[arrow]([xs,0],[xs,-60],0.02,0.8,0.02,color=black):
  qr7:=plot(-sqrt(10^2-(x-xs)^2),x=6.5..8,color=red,thickness=2):

> display(qr7,qr5,qr4,qr3,qr2,qr1,feld,view=[-50..50,-65..65]):
  # Ausschnitt s.o. links
  display(qr7,qr5,qr4,qr3,qr2,qr1,feld,view=[-22..22,-65..5]);
```

```
> # Ausgangsposition des Spielers
  z:=5;        # Abstand
  xs:=b2+z;
  ys:=0;

> alpha1:=arctan((b+z)/x);
  alpha2:=arctan(z/x);
  alpha:=unapply(piecewise(x=0,0,alpha1-alpha2),x);

> alphas:=unapply(diff(alpha(x),x),x);
  erg:=solve(alphas(x),x);
  entf:=erg[2];              # 7.848566748 m
  w:=alpha(entf);            # 0.4363483485
  evalf(w*180/Pi);           # 25.00091876 Grad
```

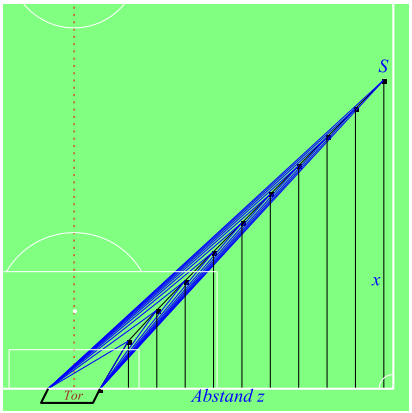
Es ist ein Extremalproblem für den optimalen Winkel α aufzustellen und zu lösen.

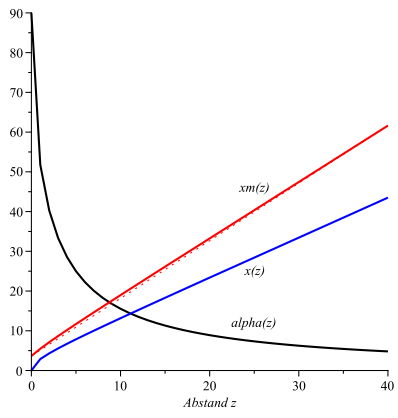


Überraschung! Die Morgenpost hat gut recherchiert. Das Ausrufezeichen hinter der Angabe $8\,m$ war berechtigt. Auch wenn die 8 Meter natürlich die Entfernung zur Torauslinie und nicht zum Tor bedeuten.

3. Spielerbewegung

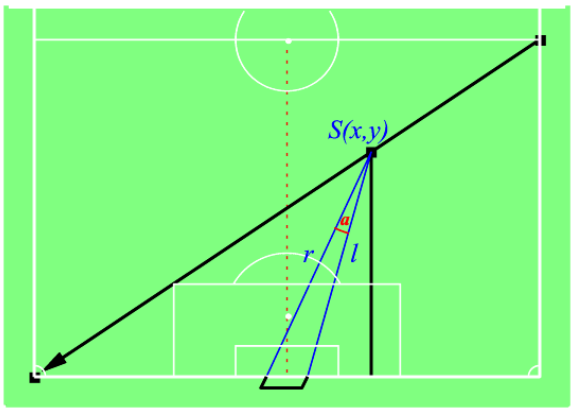
Der Spieler läuft mit Ball ab Mittellinie parallel zur Seitenlinie im Abstand von z ($z > 0$) vom Tor (Torpfosten) auf die Torauslinie zu. Bei Annäherung an das Tor verändert sich der Einschusswinkel. Es gibt einen größten. Die Bestimmung der günstigen Entfernung in Abhängigkeit vom Abstand z und Entfernung x zeigen wir in den beiden Bildern.

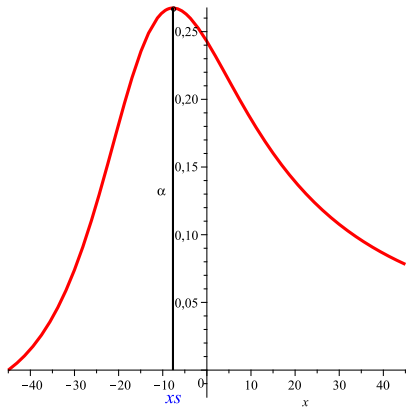




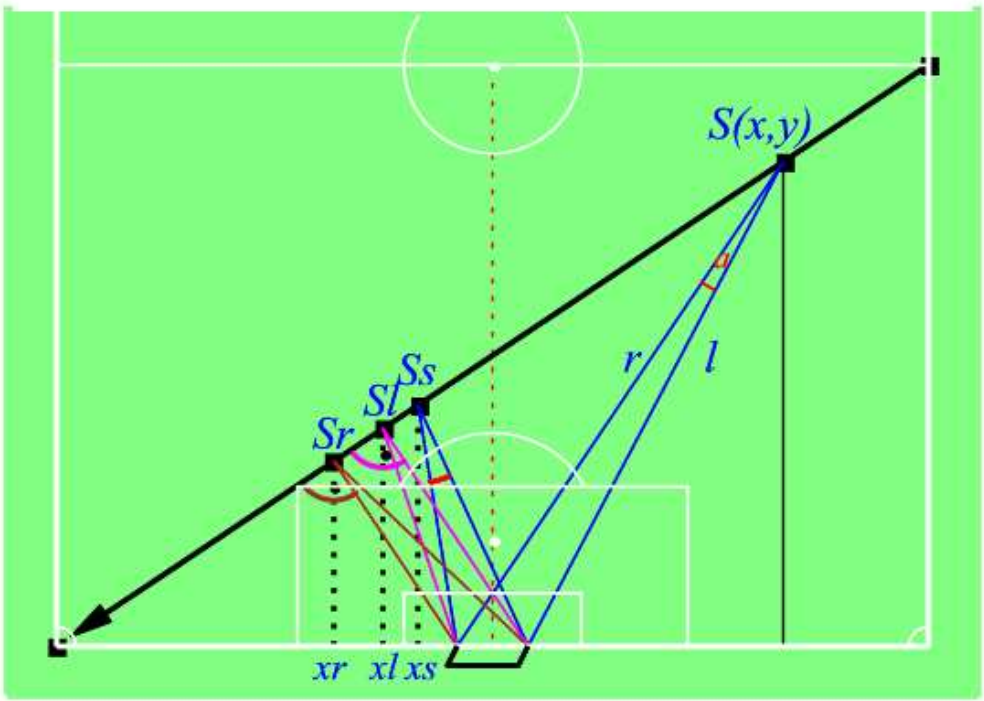
4. Spielerbewegung

Der Spieler läuft mit Ball ab Spielfeldbegrenzung (Seitenmitte = Mittellinie × Seitenlinie) in Richtung der gegenüberliegenden Eckfahne.
Bei Bewegung ändert sich die Entfernung zum Tor. Welches ist die geringste Entfernung zum linken bzw. rechten Torpfosten? Wie verändert sich der Einschusswinkel? Es gibt einen größten.





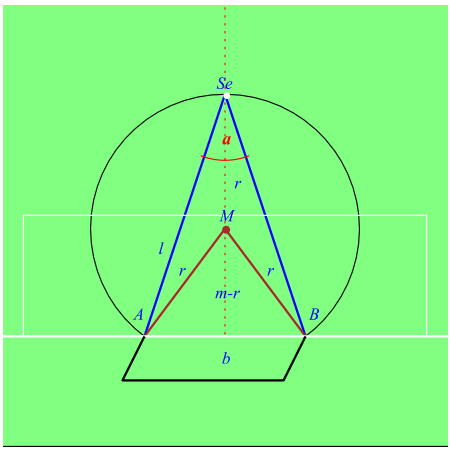
Die optimale Situation für den Einschusswinkel ist an der Stelle Ss. Der Punkt hat die Koordinate $xs = -7.68$ (Abstand nach links von Tormitte) mit der Entfernung 24.88 zur Torauslinie. Dazu gehört der optimale Winkel $\alpha = 15.32°$.
Im weiteren Verlauf kommt der Punkt Sl mit der kleinsten Entfernung zum linken sowie der Punkt Sr mit der kleinsten Entfernung zum rechten Torpfosten. Beide haben etwas kleinere Einschusswinkel.

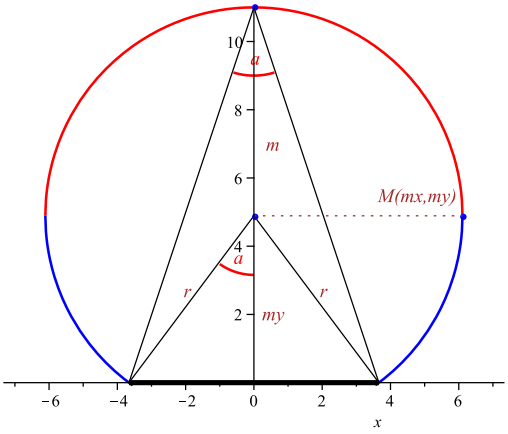


5. Spielersituation

Der Trainer möchte mit seinen Spielern möglichst effektiv den Torschuss unter dem Winkel, den ein 11-Meter-Schütze hat, trainieren. Wie sollte er die Übung organisieren? Wie sollte er die Bälle hinlegen?

Die Lösung basiert auf der Tatsache, dass alle Peripheriewinkel (Umfangswinkel) über einer Sekante im Kreis gleich sind. Die Sekante ist die Torraumlinie und mit dem Elfmeterpunkt ist der Kreis definiert, auf dem der Trainer alle Bälle für die Schussversuche ablegen kann. Natürlich ist damit auch der Kreismittelpunkt gegeben.





Man analysiere noch andere Spieler- und Spielsituationen.

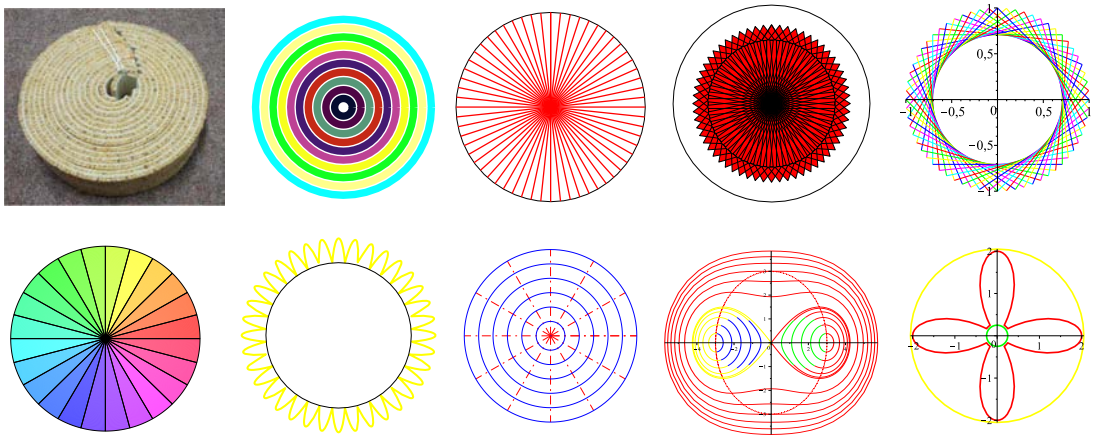
Maple-Anweisungen zur linken Abbildung

```
> # 11-Meter-Schuetze, m=11, b2=b/2, b=Torbreite
  xe:=0;
  ye:=-49;
> alfa:=2*arctan(b2/m);
  evalf(alfa*180/Pi);
  l:=b2/sin(alfa/2);
  # b2^2+(m-r)^2=r^2
  r:=(b2^2+m^2)/(2*m);

> pp67:=plot([[-b2,-60],[xe,ye],[b2,-60]],color=blue,thickness=2):
  pp68:=plot(ye-sqrt(3^2-xx^2),xx=-1.1..1.1,color=red):
  pp70:=textplot([[0,ye-2,‘ a‘]],font=[TIMES,BOLD,13],color=red):
  pp71:=plot([[-b2,-60],[0,-60+m-r],[b2,-60]],color=brown,thickness=2):
  pp72:=textplot([[xe,ye+0.5,‘Se‘],[-3,-56,‘ l‘],[0,-58,‘ m-r‘],
          [0,-61,‘ b‘],[-4,-59,‘ A‘],[4,-59,‘ B‘],[-2,-57,‘ r‘],
          [2,-57,‘ r‘],[0.5,-53,‘ r‘],[0,-54.5,‘ M‘]],
          font=[TIMES,ROMAN,12],color=blue):
  pp73:=pointplot([0,-60+m-r],
          symbol=solidcircle,symbolsize=12,color=brown):
  pp74:=plot([r*cos(t),-60+m-r+r*sin(t),t=-0.95..2*Pi-2.2],color=black):

> pph:=display(qr5,pp67,pp68,pp70,pp71,pp72,pp73,pp74):
  display(pph,feld,view=[-10..10,-65..-45]);
```

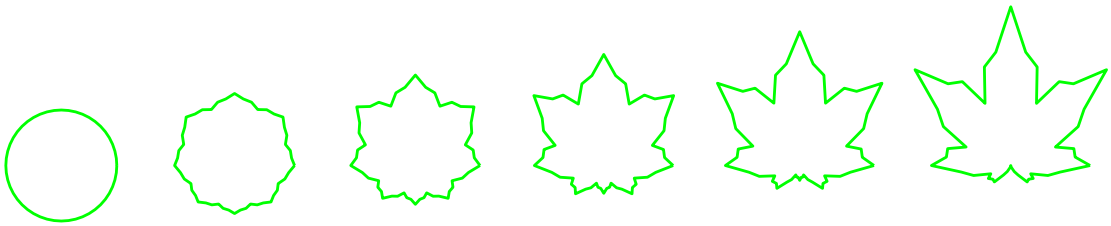
2.6.5 Runde Flächen

Verzeichnisse: `kreis_und_dreieck, kurven_2d`
Dateien: zeige_mal2.mws, kreis1,2.mws, kurven_2d.mws, animate_maple_signum.mws
Modelle: “Teppichkreis“, CD

Kreis in seiner Vielfalt



Kreis und Maple-Zeichen



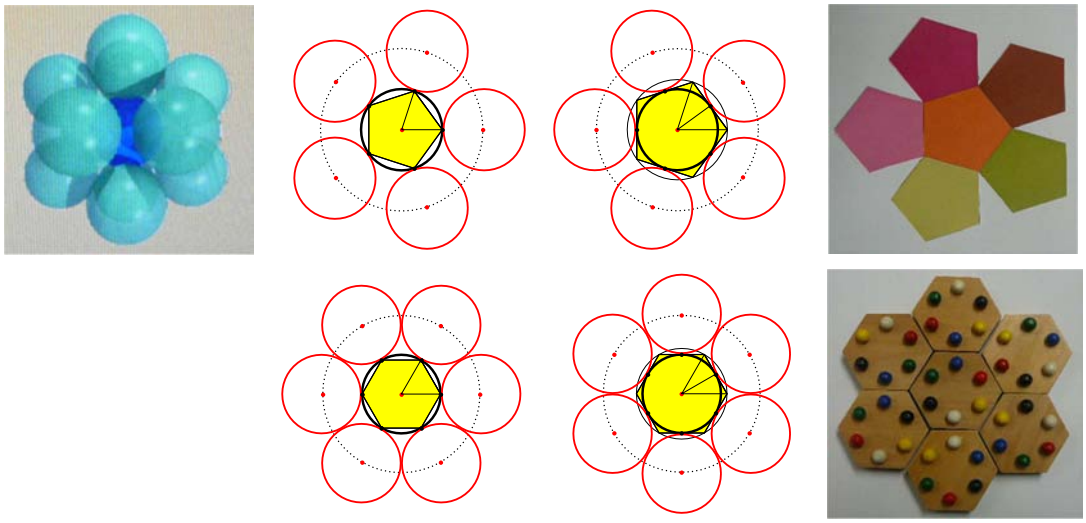
```
> # Maple sign
  X:=cos(x)*(1+k*sin(x))*(1+k*0.3*cos(8*x))*(1+k*0.1*cos(24*x)):
  Y:=sin(x)*(1+k*sin(x))*(1+k*0.3*cos(8*x))*(1+k*0.1*cos(24*x)):

> animate([X,Y,x=0..2*Pi],k=0..1,color=green,thickness=5,
          scaling=constrained,axes=none);
```

2.6.6 Kusszahlenproblem

Verzeichnis: `kreis_und_dreieck`
Dateien: kreis2.mws, PlatonischeKoerper.pdf
Modelle: Tischtennisball, Das Wabenpuzzle, Kugelhaufen

Alle Kugeln sind von gleicher Größe, davon liegt eine in der Mitte.
Kann man 13 Kugeln finden, welche die mittlere Kugel küssen und sich **nicht** gegenseitig berühren? Nein! Wie viele sind maximal möglich? 12!

Betrachten wir zunächst die Situation in der Ebene.
Alle Kreise sind von gleicher Größe, einer ist in der Mitte. Kann man 6 Kreise finden, welche den mittleren Kreis küssen und sich nicht gegenseitig berühren? Nein! Wie viele sind maximal möglich? 5!
Für den Nachweis verwenden wir das regelmäßige Fünfeck und Sechseck.
Wegen der Wabenstruktur mit dem Sechseck können es also keine 6 Kreise sein, sondern nur 5.



Für die Betrachtung im Raum nutzen wir das Dodekaeder mit seinen 12 Fünfeckseiten und kommen so auf die Zahl von 12 küssenden Kugeln.
Geht man vom Sechseck aus, dann würde man auf einer Ebene 6 Kugeln um die Mittelkugel anordnen und dann jeweils 3 Kugeln "mittig" von oben und unten anfügen. Alle äußeren Kugeln berühren dann die Mittelkugel, aber auch weitere 4 Nachbarkugeln. Herauskommt eine Haufen mit 12+1 sich berührenden Kugeln.
Eine etwas andere Anordnung der 12 äußeren Kugeln bringt jedoch die gesuchte Lösung und dabei hilft uns der Dodekaeder.

(1) Kugelgrafiken

Man teste die folgenden Maple-Anweisungen zum Plotten von Kugeln selbst.
Dabei achte man auf die Notation der zahlreichen Kommandos, die Gestaltung der Kugeloberfläche, die Gitterstruktur auf der Oberfläche, Farbgestaltung u.ä..

```
> r:=1;              # Kugelradius
  M:=[0,0,0];        # Mittelpunkt
  Ir1:=-r-1..r+1:    # Anguckbereich

> k1:=sphere(M,r):
  display(k1,scaling=constrained,style=patch,axes=boxed);
        # nur mit Standardgitter, mit Meridianen auf Oberflaeche

> k2:=sphere(M,r):
  display(k2,scaling=constrained,style=wireframe,axes=boxed);
        # nur mit Standardgitter, nur Meridiane
```

```
> k3:=implicitplot3d((x-M[1])^2+(y-M[2])^2+(z-M[3])^2=r^2,
                     x=Ir1,y=Ir1,z=Ir1,style=patch,grid=[30,30,30]):
  display(k3,axes=normal,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘]);
        # Dreiecksgitterstruktur auf Oberflaeche

> # Sphere mit festem Radius r  bzw. variablem Radius  r(theta,phi)
  plot3d(r(theta,phi),theta=a..b,phi=c..d,coords=spherical);
  k4:=plot3d(r,t=0..2*Pi,p=0..Pi,coords=spherical,view=[Ir1,Ir1,Ir1]):
  display(k4,axes=normal,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘]);
        # mit Meridianen auf Oberflaeche
  display(k4,axes=normal,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘],
          transparency=0.5);
  display(k4,axes=normal,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘],
          style=wireframe,color=gray);

> # implicitplot3d(f(r,theta,phi),r=r0..r1,theta=a..b,phi=c..d,
  #                coords=spherical);   d.h. f(r,theta,phi)=0
  k5:=implicitplot3d(t=r,t=0..r,x=0..2*Pi,y=0..Pi,coords=spherical,
                     grid=[20,20,20]):
  display(k5,axes=boxed,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘]);
        # keine Kugelerkennung im gewaehlten Gitter

> k6:=implicitplot3d(t=r,t=0..r,x=0..2*Pi,y=0..Pi,coords=spherical,
                     grid=[21,21,21]):
  display(k6,axes=boxed,scaling=constrained,labels=[‘ x‘,‘ y‘,‘ z‘]);
        # mit Meridianen und Dreiecken auf Oberflaeche
```

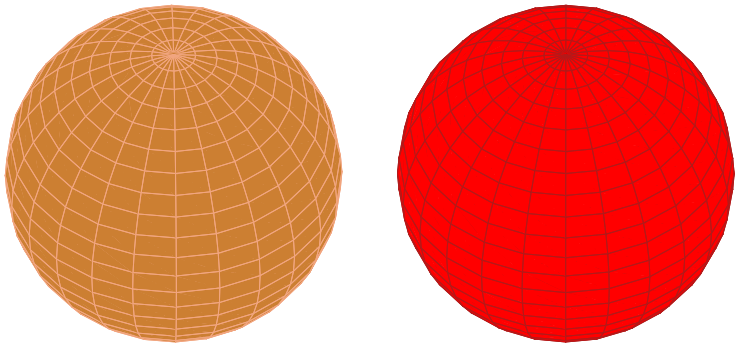
(2) 13 Kugeln mit Sechseck

Wir verwenden 3 Schichten von Kugeln: in der Mitte 7 Kugeln, unten und oben jeweils 3 Kugeln. Die Mittelpunkte der Kugeln sind einfach zu berechnen.

```
> r:=1:          # Kugelradius
                 # Mitten von 3 aneinanderliegenden Kugeln bilden Dreieck
                 # 2r = Dreiecksseite
  h:=r*sqrt(3): # Dreieckshoehe
  h3:=h/3: s:=2*h/3:
  n:=12:         # Anzahl der Aussenkugeln
  M:=[0,0,0]:    # Mittelpunkt der Zentrumskugel
  Ir:=-r..r:
  Ir1:=-r-1..r+1:

> # Mittelpunkte der 12 beruehrenden Kugeln
  M1:=[2*r,0,0]:  M2:=[r,h,0]:    M3:=[-r,h,0]:
  M4:=[-2*r,0,0]: M5:=[-r,-h,0]:  M6:=[r,-h,0]:
  U1:=[r,h3,-h]:  U2:=[-r,h3,-h]: U3:=[0,-s,-h]:
  O1:=[r,h3,h]:   O2:=[-r,h3,h]:  O3:=[0,-s,h]:
```

Ein Test zu Kommandos für Kugeln ergibt günstiges Oberflächenmuster, Gitter und Farben. Für die Kugeloptik ist es vorteilhaft, wenn auf der Oberfäche die Meridiankurven eingezeichnet werden. Man achte auch auf die Verwendung der Optionen in den Kommandos.

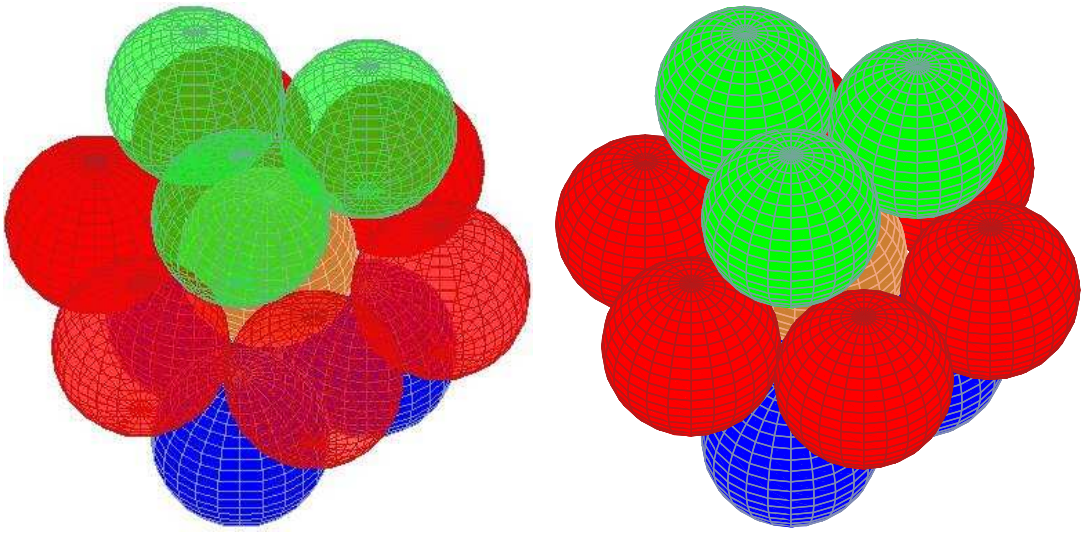
```
> km1:=sphere(M,r,scaling=constrained,style=patchnogrid,color=gold):
  macro(farbe26=COLOR(RGB,0.95,0.65,0.50)):  # beige, dunkel
  #km2:=sphere(M,r,style=wireframe):         # Problem mit wireframe
                                             # Umweg
  km2:=display(sphere(M,r),style=wireframe,color=farbe26):
               # nur mit Standardgitter, mit Meridianen
  display(km1,km2);
  # analog mit rot
```



Beim Kugelhaufen geben wir den Schichten verschiedene Farben und machen die vorne liegenden Kugeln etwas transparent.

```
> # Farben fuer Meridiane
  macro(farbe26=COLOR(RGB,0.95,0.65,0.50)):  # beige, dunkel
  macro(farbe32=COLOR(RGB,0.70,0.10,0.10)):  # rot, dunkel
  macro(farbe20=COLOR(RGB,0.50,0.55,0.70)):  # stahl
  macro(farbe14=COLOR(RGB,0.45,0.65,0.6)):   # gruen, dunkel

> # Mitte gold
  km1:=sphere(M,r,scaling=constrained,style=patchnogrid,color=gold):
  km2:=display(sphere(M,r),style=wireframe,color=farbe26):

> # 6 mittlere Kugeln rot
  k61:=sphere(M6,r,style=patchnogrid,color=red,transparency=0):
  k62:=display(sphere(M6,r),style=wireframe,color=farbe32):
  k6:=display(k61,k62):
  k51:=sphere(M5,r,style=patchnogrid,color=red,transparency=0):
  k52:=display(sphere(M5,r),style=wireframe,color=farbe32):
  k5:=display(k51,k52):
  k41:=sphere(M4,r,style=patchnogrid,color=red,transparency=0):
  k42:=display(sphere(M4,r),style=wireframe,color=farbe32):
  k4:=display(k41,k42):
  k31:=sphere(M3,r,style=patchnogrid,color=red,transparency=0.5):
  k32:=display(sphere(M3,r),style=wireframe,color=farbe32):
  k3:=display(k31,k32):
  k21:=sphere(M2,r,style=patchnogrid,color=red,transparency=0.6):
  k22:=display(sphere(M2,r),style=wireframe,color=farbe32):
  k2:=display(k21,k22):
  k11:=sphere(M1,r,style=patchnogrid,color=red,transparency=0.5):
  k12:=display(sphere(M1,r),style=wireframe,color=farbe32):
  k1:=display(k11,k12):
```

```
> # 3 untere Kugeln blau
  k71:=sphere(U1,r,style=patchnogrid,color=blue,transparency=0.0):
  k72:=display(sphere(U1,r),style=wireframe,color=farbe20):
  k7:=display(k71,k72):
  k81:=sphere(U2,r,style=patchnogrid,color=blue,transparency=0.0):
  k82:=display(sphere(U2,r),style=wireframe,color=farbe20):
  k8:=display(k81,k82):
  k91:=sphere(U3,r,style=patchnogrid,color=blue,transparency=0.0):
  k92:=display(sphere(U3,r),style=wireframe,color=farbe20):
  k9:=display(k91,k92):

> # 3 obere Kugeln gruen
  k101:=sphere(O1,r,style=patchnogrid,color=green,transparency=0.6):
  k102:=display(sphere(O1,r),style=wireframe,color=farbe14):
  k10:=display(k101,k102):
  k111:=sphere(O2,r,style=patchnogrid,color=green,transparency=0.6):
  k112:=display(sphere(O2,r),style=wireframe,color=farbe14):
  k11:=display(k111,k112):
  k121:=sphere(O3,r,style=patchnogrid,color=green,transparency=0.6):
  k122:=display(sphere(O3,r),style=wireframe,color=farbe14):
  k12:=display(k121,k122):

> # Reihenfolge beachten
  display(k7,k8,k9,k6,k5,k4,km1,km3,k1,k2,k3,k10,k11,k12,
          scaling=constrained);
```



Im Kugelhaufen haben die versteckten (hintere und untere) Außenkugeln keine Durchsichtigkeit gemäß `transparency=0`, hingegen sollen die vorne liegende Kugeln transparent sein.
Beim Maple-Export des Bildes als ps- oder eps-Datei geht die Transparenz verloren (siehe rechtes Bild). Mit dem Umweg über Export als jpeg-Datei und dann erst der Umwandlung mittels eines Grafik-Programms in eine Postscript-Datei kann man die Transparenz retten.

(3) 13 Kugeln mit Fünfeck

Wir verwenden 5 Schichten von Kugeln: in der Mitte die Zentrumskugel, in der unteren und oberen Mittelschicht jeweils 5 Kugeln sowie unten und oben jeweils 1 Kugel.
Die Mittelpunkte der Kugeln sind nicht ganz so einfach zu berechnen. Aber ein kleiner Trick hilft uns da weiter. Im Maple-Paket `geom3d` kann man für den Dodekaeder das Kommando `dodecahedron` verwenden. Es liefert neben der Grafik auch noch zahlreiche nützliche Parameter und Informationen über den Dodekaeder, aus denen sich die Miitelpunkte leicht ableiten lassen.

```
> with(geom3d):
> dodecahedron(th,point(o,0,0,0),1):   # Mitte M=[0,0,0]
                                       # Radius der Aussenkugel r=1
  draw(th);

> radius(th);            # Radius der Aussenkugel
  form(th);              # Typ des Polyeders
  schlafli(th);
  center(th);            # Schwerpunkt, Mitte
  M:=[0,0,0];

> ri:=InRadius(th);      # Radius der Innenkugel  ri=0.7946544727
  ri:=simplify(ri);
  evalf(ri);
  rih:=sqrt(1+2/sqrt(5))/sqrt(3);   # Vereinfachung per Hand
  evalf(rih);

  ra:=radius(th);        # Radius der Aussenkugel  ra=1
  s:=sides(th);          # Kantenlaenge
  s:=simplify(s);
  e:=MidRadius(th);      # Entfernung von Koerpermitte zu Kante
  e:=simplify(e);        # Mittelradius
  evalf(e);
  V:=volume(th);         # Volumen
  A:=area(th);           # Oberflaeche
  Seiten:=faces(th):     # 12 Seiten, je Seite 5 Ecken
  Ecken:=vertices(th):   # 20 Dodekaederecken
```

Möchten wir nun den Innenradius des Dodekaeders zu 1 machen, brauchen nur den Außenradius proportional zu vergrößern und notieren einfach den folgenden Aufruf.

```
> dodecahedron(th1,point(o,0,0,0),1/ri):   # Mitte M=[0,0,0]
                                           # Radius der Aussenkugel >1
                                           # Radius der Aussenkugel =1

> radius(th1):           # Radius der Aussenkugel
  evalf(%);
  form(th1);             # Typ des Polyeders
  schlafli(th1);
  center(th1);           # Schwerpunkt, Mitte
  M:=[0,0,0]:
```

1.258408573
dodecahedron3d
[5, 3]
o

```
> ri1:=InRadius(th1):    # Radius der Innenkugel
  ri1:=simplify(ri1);
  evalf(ri1);
  ra1:=radius(th1):      # Radius der Aussenkugel
  evalf(ra1);
  s1:=sides(th1):        # Kantenlaenge
  s1:=simplify(s1):
  evalf(s1);
  e1:=MidRadius(th1):    # Entfernung von Koerpermitte zu Kante
  e1:=simplify(e1):      # Mittelradius
  evalf(e1);
  V1:=volume(th1):       # Volumen
  A1:=area(th1):         # Oberflaeche

> Seiten1:=faces(th1):   # 12 Seiten, je Seite 5 Ecken
  Ecken1:=vertices(th1): # 20 Dodekaederecken
```

$ri1 := 1$
1.
1.258408573
0.8980559536
1.175570504

Nun erfolgt die Bestimmung der 12 Mittelpunkte M_i der 12 Seiten.
$2 \cdot M_i$ sind dann die Mitten der 12 berührenden Kugeln mit dem Radius 1.

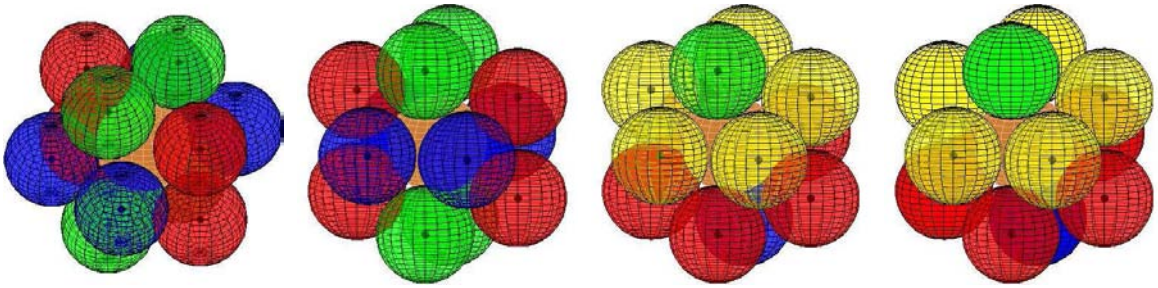
```
> # Mittelpunkte als konvexe Kombination der Seitenecken
Mh:=array(1..n,[]):
k:='k':
for l from 1 to n do
  Mh[l]:=2*evalf(1/5*sum(Seiten1[l][k],k=1..5));
end do:
# je 4 Mittelpnkte liegen in einer Koordinatenebene
# als Ecken eines Rechtecks
evalm(Mh);
```

```
[[0.,1.701301619,1.051462226],[0.,1.701301619,-1.051462226],[0.,-1.701301619,1.051462226],[0.,-1.701301619,-1.051462226],
 [1.051462226,0.,1.701301619],[1.051462226,0.,-1.701301619],[-1.051462226,0.,1.701301619],[-1.051462226,0.,-1.701301619],
 [1.701301619,1.051462226,0.],[1.701301619,-1.051462226,0.],[-1.701301619,1.051462226,0.],[-1.701301619,-1.051462226,0.]]
```

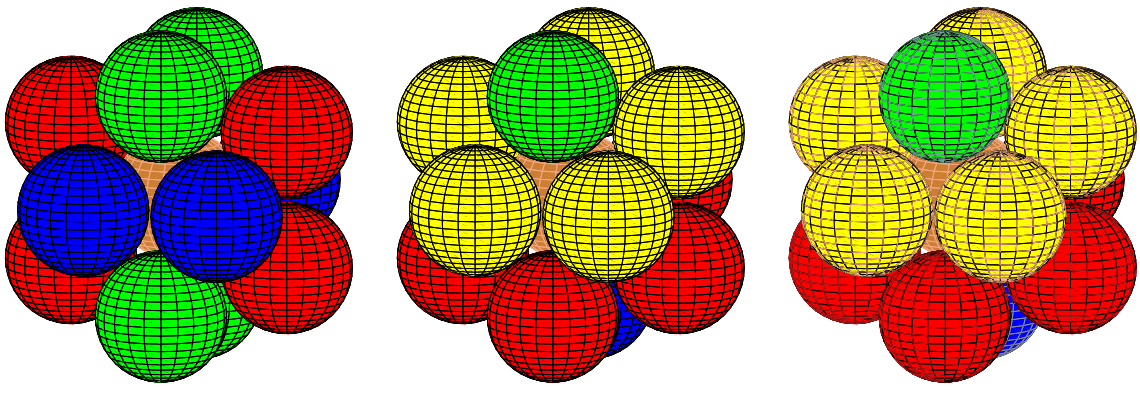
In beiden Paketen `geom3d` und `plottools` gibt es das Kommando `sphere`, aber mit unterschiedlichem Parameterkonzept. Somit ist zwecks Eindeutigkeit der Zuordnung dann der Auruf `plottools[sphere](...)` zu verwenden.

```
> # Zentrumskugel, gold
  macro(farbe26=COLOR(RGB,0.95,0.65,0.50)):  # beige, dunkel
  km1:=plottools[sphere](M,ri1,style=patchnogrid,color=gold):
  km2:=display(plottools[sphere](M,ri1),style=wireframe,color=farbe26):
  km:=display(km1,km2,scaling=constrained):
```

```
> with(plottools):
> sp:=array(1..n,[]):
  # Kugelmittelpunkte
  pp:=pointplot3d([M,seq(Mh[l],l=1..n)],
                  symbol=solidcircle,symbolsize=10,color=black):
  # 12 Kugeln
  for l from 1 to 4 do
    sp[l]:=sphere(Mh[l],ri1,style=patch,color=red,transparency=0.5):
  end do:
  for l from 5 to 8 do
    sp[l]:=sphere(Mh[l],ri1,style=patch,color=green,transparency=0.5):
  end do:
  for l from 9 to n do
    sp[l]:=sphere(Mh[l],ri1,style=patch,color=blue,transparency=0.5):
  end do:

> # Bild 1
  display(km,pp,seq(sp[l],l=1..n),scaling=constrained);
> # Bild 2 nach Drehung
  # Bild 3 nach Umfaerbung
  # Bild 4 nach Verschoenerung jeweils
  display(pp,km,seq(sp[l],l=1..n),scaling=constrained,orientation=[15,80]);
```



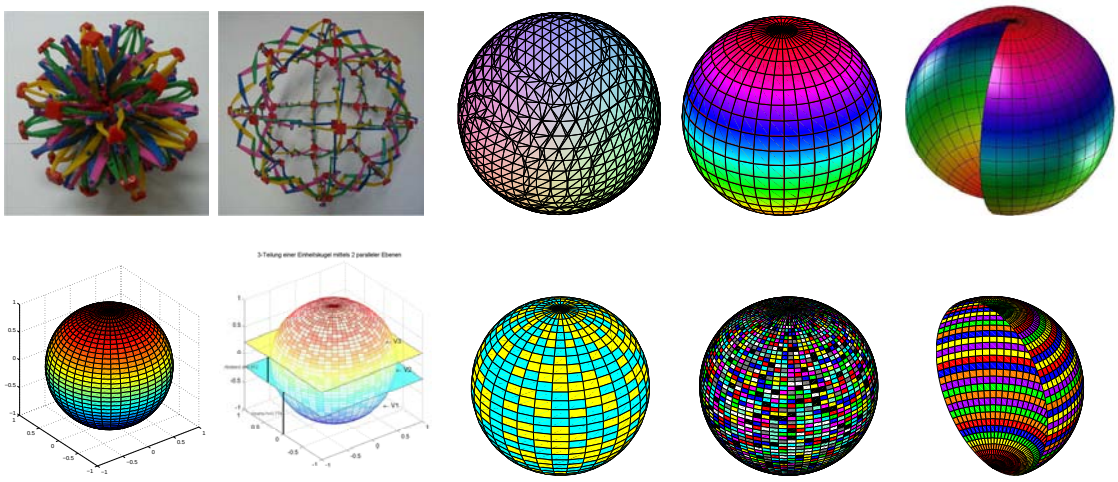
In den ersten drei Kugelhaufen haben alle 12 Außenkugeln die Durchsichtigkeit `transparency=0.5`, im letzten Bild nur die vorne liegenden Kugeln.
Mit dem Umweg über den Export als jpeg-Datei und dann erst der Umwandlung mittels eines Grafik-Programms in eine Postscript-Datei kann man die Transparenz retten. Aber beim Maple-Export der Bilder als ps- oder eps-Dateien geht die Transparenz verloren (s. unten zu den Bildern 2,3,4) und es treten ev. weitere Effekte auf.



2.6.7 Runde Körper

Verzeichnisse: `flaechen_3d\kugel, zahlenwuerfel`
Dateien: zeige_mal2.mws, zahlenwuerfel1.mws, kugel1,2.m
Modelle: Ball, Fußball, Tischtennisball, Kugelgitter

Kugel in ihrer Vielfalt



Kugeln mit MATLAB

Kugeln Nr. 4-7 (aus kugel1.m), Nr. 8-10 (aus kugel2.m) zu obiger Abbildung

```
% kugel1.m

diary kugel1.txt          % Protokolldatei
diary off
echo off
clear all
clc
clf

format compact
format short

% Nr. 4
% Kugel, komplett
phi = 0:pi/20:2*pi;
theta = 0:pi/20:pi;
[U,V] = meshgrid(phi,theta);
X = sin(V).*cos(U);
Y = sin(V).*sin(U);
Z = cos(V);
```

```
figure(1) ;
h = surf(X,Y,Z,Z);   % Farbwerte C=Z
axis equal
axis off
colormap(hsv(1024));
shading interp
set(h,'EdgeColor','k')
% moeglicher Grafikexport
% print -dpsc 'kugel1_01.ps'     bzw.  print('-dpsc','kugel1_01.ps')
% print -djpeg 'kugel1_01.jpg'   bzw.  print('-djpeg','kugel1_01.jpg')
pause

% Nr. 5
% Kugel, ausgeschnitten
phi_a = 6/5*pi:pi/20:14/5*pi;
theta = 0:pi/20:pi;
[U,V] = meshgrid(phi_a,theta);
X = sin(V).*cos(U);
Y = sin(V).*sin(U);
Z = cos(V);

figure(2) ;
h = surf(X,Y,Z,Z);   % Farbwerte C=Z
axis equal
axis off
colormap(hsv(1024))
shading interp
material shiny
lighting gouraud
lightangle(80,-40)
lightangle(-90,60)
set(h,'EdgeColor','k')
pause

% Nr. 6
% Kommando sphere fuer Einheitskreis
figure(3)
N = 40;
[X,Y,Z] = sphere(N);
surf(X,Y,Z)
colormap jet
axis equal
pause
```

```
% Nr. 7
% Geometrie: 3-Teilung einer Einheitskugel mittels 2 paralleler Ebenen
figure(12)
[X,Y,Z] = sphere(50);          % hoehere Aufloesung
mesh(X,Y,Z,'FaceAlpha',0.8)
axis on
axis square
axis equal
axis([-1 1 -1 1 -1 1])
title('3-Teilung einer Einheitskugel mittels 2 paralleler Ebenen')
hold on
surf(X,Y,Z,'EdgeColor','none','FaceAlpha',0.3)

x1 = linspace(-1,1,2);
y1 = linspace(-1,1,2);
z1 = 0.226*ones(2,2);
surf(x1,y1,z1,'FaceAlpha',0.5)
z2=-0.226*ones(2,2);
surf(x1,y1,z2,'FaceAlpha',0.6)
k = 2;
x = linspace(-1,-1,k);
y = linspace(0.4,0.4,k);
z = linspace(-0.226,0.226,k);
plot3(x,y,z,'LineWidth',2,'Color','k')
text(-1.4,1,-0.1,'Abstand d=0.452','FontSize',8)
k = 2;
x = linspace(-1,-1,k);
y = linspace(0,0,k);
z = linspace(-0.226,-1,k);
plot3(x,y,z,'LineWidth',2,'Color','k')
text(-1.2,0.55,-0.8,'Hoehe h=0.774','FontSize',8)
text(0.6,-0.5,-0.8,'\leftarrow V1','FontSize',11)
text(0.7,-0.7,-0.1,'\leftarrow V2','FontSize',11)
text(0.5,-0.7,0.5,'\leftarrow V3','FontSize',11)
hold off
pause

%-------------------------------------------------------------
% kugel2.m

% Nr. 8
% [x,y,z]=sphere(n) generates three (n+1)-by-(n+1) matrices,
% so that surf(x,y,z) produces a unit sphere
% Sphere with surface color from a Hadamard matrix
figure(37)
k = 5;
```

```
n = 2^k-1;
[x,y,z] = sphere(n);
c = hadamard(2^k);            % Matrix mit 1,-1 -> 2 Farben
surf(x,y,z,c);
colormap([1  1  0; 0  1  1])  % gelb, cyan
axis off
pause

% Nr. 9
% Sphere with surface color from a random matrix
figure(38)
k = 6;
n = 2^k-1;
[x,y,z] = sphere(n);
c = fix(10*rand(2^k));        % Matrix mit 0,1,...,9 -> 10 Farben
surf(x,y,z,c);
colormap([1 1 1               % white
          1 1 0               % yellow
          1 0 1               % magenta
          1 0 0               % red
          0 1 1               % cyan
          0 1 0               % green
          0 0 1               % blue
          127/255 1 212/255   % aquamarine
          0.5 0.5 0.5         % gray
          0 0 0 ])            % black
                              % pink [1 0.4 0.6]
axis off
pause

% Nr. 10
% Halbkugel
figure(34)
phi = -pi/2:pi/40:pi/2;
theta = -pi/2:pi/40:pi/2;
[U,V] = meshgrid(phi,theta);
X = cos(V).*cos(U);
Y = cos(V).*sin(U);
Z = sin(V);
figure(33)
h = surf(X,Y,Z);     % Farbwerte C=Z
axis equal
axis off
colormap prism
pause
```

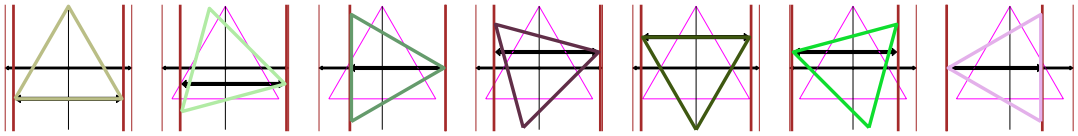
2.6.8 Flächen nicht gleicher und gleicher Dicke

Verzeichnis: `solids_pseudokugeln`
Dateien: rotation_dreieck1.mws, rotation_quadrat1.mws,
rotation_dreieck_rund1,2,3.mws, bewegen_dreieck_rund1.mws
Modelle: Schnittmuster, Fahrgestell

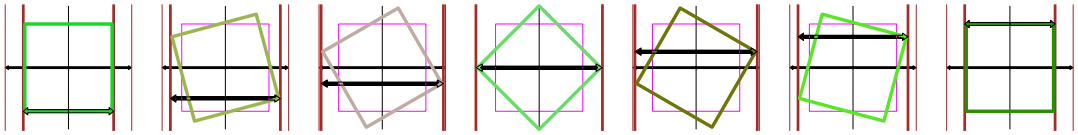
Flächen nicht gleicher Dicke

Wir lassen ein Polygon um seinen Schwerpunkt bzw. seine Mitte rotieren und beobachten seine Dicke durch den Abstand von begrenzenden parallelen Geraden.

Gleichseitiges Dreieck

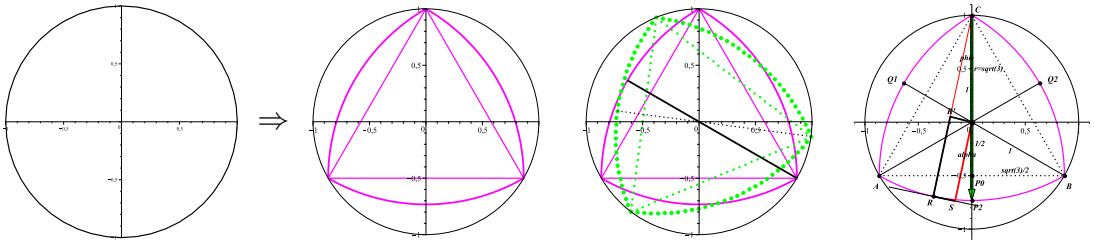


Quadrat



Flächen gleicher Dicke

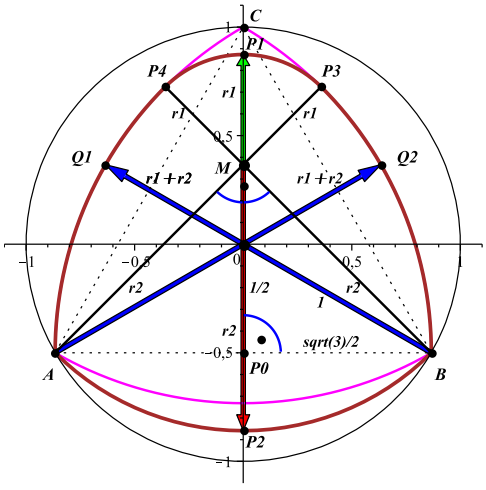
Vom Kreis zum abgerundeten Dreieck, Konstruktion



Weitere Abrundung des Dreiecks

Konstruktion einer Fläche gleicher Dicke
Nennen wir es einfach **Zweieck**.

Seine Dicke ist die Summe $r = r_1 + r_2$.
Die Randkurve setzt sich aus 4 Kreisbögen AB ($MA = MP_2 = MB = r_2$), BP_3, P_3P_4 ($MP_3 = MP_1 = MP_4 = r_1$), P_4A zusammen. An den Punkten P_3 und P_4 haben wir glatte Übergänge. Die Strecken AP_3 und BP_4 mit der Länge r schneiden sich im Punkt M und stehen hier senkrecht aufeinander, das muss i. Allg. nicht sein. Meist lässt man die Figur um den Schwerpunkt des Dreiecks 0 rotieren.



Wichtige Elemente bei der Betrachtung der Fläche sind die Beschreibung der Kurve für den Flächenrand sowie die Rotation in der Ebene eines Punktes $P(x, y)$ bzw. einer Kurve mit der Parameterdarstellung $D(t) = (x(t), y(t)),\ 0 \le t \le 2\pi\ (t \sim \varphi)$ um den Koordinatenursprung. Es sei $D(0) = D(2\pi) = A,\ D(2\pi/3) = B,\ D(4\pi/3) = C$.

Gleichseitiges Dreieck als Kurve, $\rho > 0$ Außenkreisradius

$$D(t) = \rho \begin{cases} \Big(-\frac{\sqrt{3}}{2} + \frac{3\sqrt{3}}{2\pi}\,(t-0), -1/2 \Big), & \text{falls } \ 0 \le t < \frac{2\pi}{3}, \\ \Big(\frac{\sqrt{3}}{2} - \frac{3\sqrt{3}}{4\pi}\,(t-\frac{2\pi}{3}), -\frac{1}{2} + \frac{9}{4\pi}\,(t-\frac{2\pi}{3}) \Big), & \text{falls } \ \frac{2\pi}{3} \le t < \frac{4\pi}{3}, \\ \Big(0 - \frac{3\sqrt{3}}{4\pi}\,(t-\frac{4\pi}{3}), 1 - \frac{9}{4\pi}\,(t-\frac{4\pi}{3}) \Big), & \text{falls } \ \frac{4\pi}{3} \le t < 2\pi. \end{cases}$$

```
> # Dreieck
  dreieck1:=unapply(piecewise(
         phi<2*Pi/3,rho*[-sqrt(3)/2+3*sqrt(3)/(2*Pi)*(phi-0), -1/2],
         phi<4*Pi/3,rho*[sqrt(3)/2-3*sqrt(3)/(4*Pi)*(phi-2*Pi/3),
                         -1/2+9/(4*Pi)*(phi-2*Pi/3)],
                    rho*[0-3*sqrt(3)/(4*Pi)*(phi-4*Pi/3),
                         1-9/(4*Pi)*(phi-4*Pi/3)]),phi);
```

Abgerundetes gleichseitiges Dreieck als Kurve, $\rho > 0$ Außenkreisradius

$$DR(t) = \rho \begin{cases} \Big(0 + \sqrt{3}\cos(\frac{4\pi}{3} + \frac{1}{2}\,(t-0)), 1 + \sqrt{3}\sin(\frac{4\pi}{3} + \frac{1}{2}\,(t-0)) \Big), & 0 \le t < \frac{2\pi}{3}, \\ \Big(-\frac{\sqrt{3}}{2} + \sqrt{3}\cos(\frac{1}{2}\,(t-\frac{2\pi}{3})), -\frac{1}{2} + \sqrt{3}\sin(\frac{1}{2}\,(t-\frac{2\pi}{3})) \Big), & \frac{2\pi}{3} \le t < \frac{4\pi}{3}, \\ \Big(\frac{\sqrt{3}}{2} + \sqrt{3}\cos(\frac{1}{2}\,t), -\frac{1}{2} + \sqrt{3}\sin(\frac{1}{2}\,t) \Big), & \frac{4\pi}{3} \le t < 2\pi. \end{cases}$$

```
> # Abgerundetes Dreieck
  dreieck_rund1:=unapply(piecewise(
         phi<2*Pi/3,rho*[0+sqrt(3)*cos(4*Pi/3+phi/2),
                         1+sqrt(3)*sin(4*Pi/3+phi/2)],
         phi<4*Pi/3,rho*[-sqrt(3)/2+sqrt(3)*cos((phi-2*Pi/3)/2),
                         -1/2+sqrt(3)*sin((phi-2*Pi/3)/2)],
                    rho*[sqrt(3)/2+sqrt(3)*cos(phi/2),
                         -1/2+sqrt(3)*sin(phi/2)]),phi);
```

Ebene Rotation eines Vektors $r = (r_1, r_2)^T$ um den Winkel α im mathematisch postiven Sinn mit der orthonormalen Rotationsmatrix

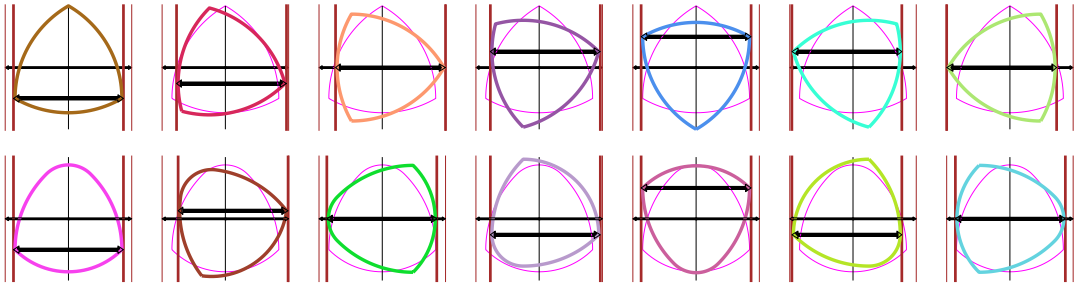
$$R = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}, \quad R^T = R^{-1}.$$

```
> # Prozedur fuer Rotation
  rotiere:=(r,alpha)->[cos(alpha)*r[1]-sin(alpha)*r[2],
                       sin(alpha)*r[1]+cos(alpha)*r[2]];
```

Rotation einer Kurve, d.h. Rotation einer Liste von Punkten, welche die Kurve gut beschreibt, unter Verwendung der Maple-Anweisung `map`.

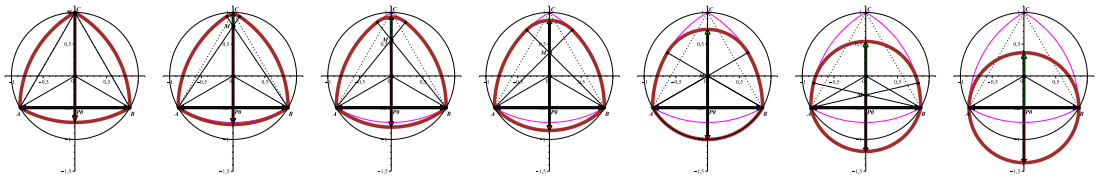
```
> n:=90:   # Feinheit der Kurve, bei Dreieck reichen 3 Punkte
  dtn:=2*Pi/n:
  rho:=1:

> # Kurve: Kreis, Dreieck, abgerundetes Dreieck
  liste:=[seq(dreieck1(i*dtn),i=0..n)]:
  alpha:=Pi/8:
  liste2:=map(rotiere,liste,alpha):
```

Dickenmessung der Figur mittels Abstand paralleler vertikaler Geraden



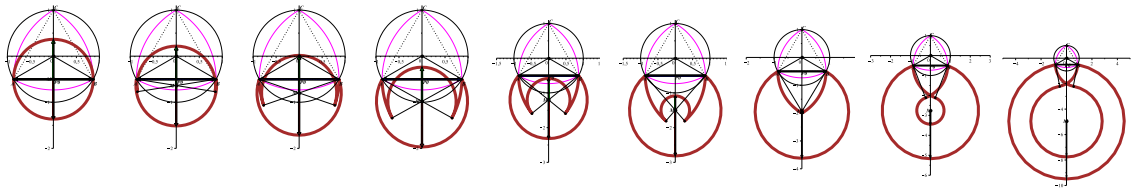
Vom abgerundeten Dreieck über das Zweieck zum Kreis

Bei Zweieck haben wir festgestellt, dass die Sekanten AP_3 und BP_4, die sich im Punkt M schneiden, nicht unbedingt senkrecht aufeinander stehen müssen. Deshalb gibt es zwischen den Grenzfällen des abgerundeten Dreiecks bei $M = C$ und des Kreises bei $M = P_0$ unendlich viele Zweiecke gleicher Dicke.



Fortsetzung der Idee des Bewegungsablaufs mit dem letzten Kreis zu 2 Kreisen

Es ist schon interessant, wie man von einem Dreieck über "Umwege" zu zwei Kreisen gelangt. Was passiert, wenn man den Prozess weiter verfolgt? Lassen wir also den Punkt M in Richtung P_2 und weiter bewegen. Es entstehen zwei horizontale Geraden durch die Punkte A und $A - (0, \sqrt{3})$.



Die Framebereiche wurden angepasst.

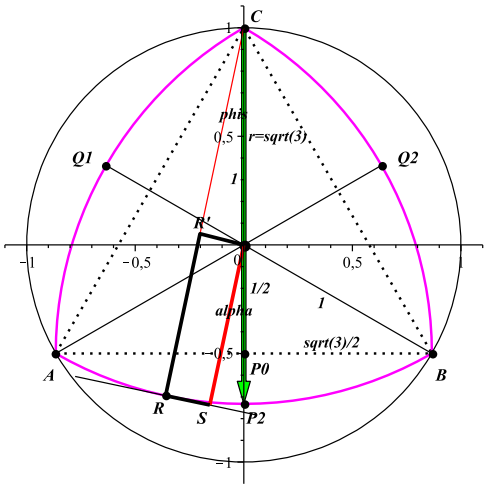
Das abgerundete Dreieck mit gleicher Dicke wollen wir nun als Rad verwenden und bilden damit eine Achse mit 2 Rädern. Da diese auf einer Ebene laufen soll, müssen wir die Bewegung des Drehpunktes/Schwerpunktes M des Dreiecks simulieren, insbesondere die periodische Veränderung in der Vertikalen.

Die Abstandsfunktion $(0,S)$ bewegt sich von kleinsten Länge $(0,P_2) = \sqrt{3}-1$ bis größten $(0,A) = 1$, mit periodischer Wiederholung. Man nutzt die Beziehungen im Dreieck $(0,P_2,S)$ und Rechteck $(0,S,R,R')$ mit den Winkeln und Seiten.

Der Winkelbereich ist $0 \le \alpha \le \pi/3$.

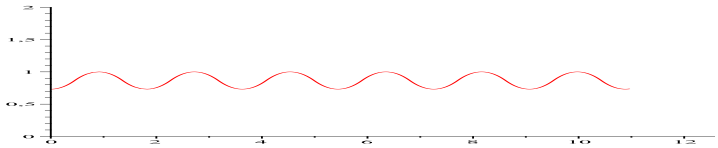
Sinnvoll ist eine Fallunterscheidung:
$0 \le \alpha \le \pi/6$ (siehe Grafik) und
$\pi/6 \le \alpha \le \pi/3$ (einfachere Berechnung).

Die Beschreibung der Randkurve erfolgt mit $DR(t),\ 0 \le t \le 2\pi,\ \rho = 1$.



Wegen der auftretenden Periode erzeugen wir die Abstandsfunktion `ym` schrittweise.

```
> # Schritt 1
  ym1:=alpha->piecewise(alpha<=Pi/6,
                sqrt(dreieck_rund1(alpha)[1]^2+
                (1-dreieck_rund1(alpha)[2])^2)-cos(alpha),
                sin(Pi/6+alpha));
  T:=sqrt(3)*Pi;
  xm:=alpha->T/(2*Pi)*alpha;

  # Schritt 2, Spiegelung, 0<=alpha<=2Pi/3
  ym2:=alpha->piecewise(alpha<=Pi/3,ym1(alpha),ym1(2*Pi/3-alpha));

> # Schritt 3, Wiederholung, periodische Fortsetzung
  ym:=alpha->ym2(alpha-2*Pi/3*floor(alpha*3/(2*Pi)));

> # Test, Mittelpunktskurve
  plot([xm(t),ym(t),t=0..4*Pi+0.1],
       scaling=unconstrained,view=[0..4*Pi,0..2]);
```

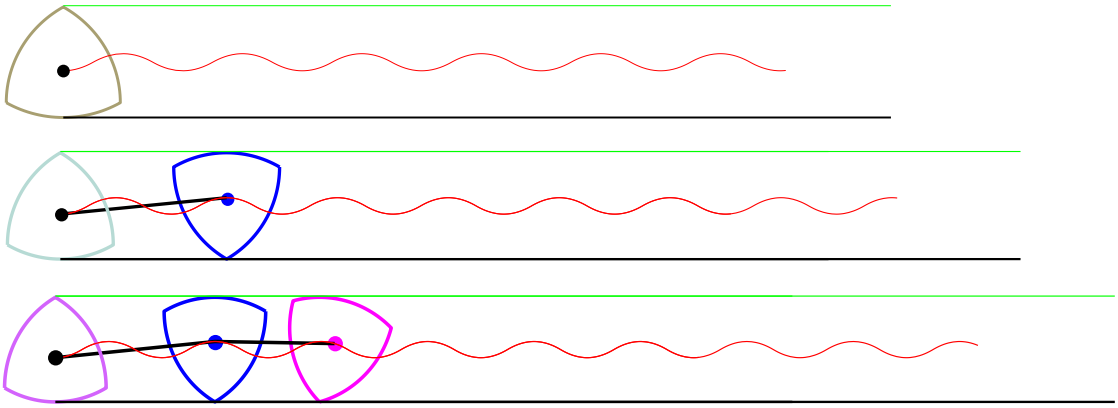


Zu Animation der Bewegung müssen wir die Koordinatenfunktionen `xm, ym` der Bewegung als Mittelpunkte mit der Rotation noch kombinieren.

```
> rotiere1:=(r,alpha)->[xm(alpha)+cos(alpha)*r[1]+sin(alpha)*r[2],
                        ym(alpha)-sin(alpha)*r[1]+cos(alpha)*r[2]];
```

Bei mehreren Achsen braucht man nur noch den Achsenabstand zu berücksichtigen.

Ein-, Zwei- und Dreiachser mit Dreiecksrädern



Modell eines Fahrgestells

Zusatzaufgabe

Da wir einmal bei Bewegungsabläufen mit Fahrgestellen sind, wird noch folgendes kleine Problem betrachtet.
Wir wollen eine Fahrt mit einem Auto auf Rädern machen, die Quadrate sind.
Wie ist bei quadratischen Rädern die Fahrbahn zu modellieren, dass unser Auto nicht schaukelt, d.h. damit die Achsen des Autos horizontal auf gleicher Höhe bleiben?
Natürlich lassen wir auch den Vergleich mit einem richtigen Auto zu.

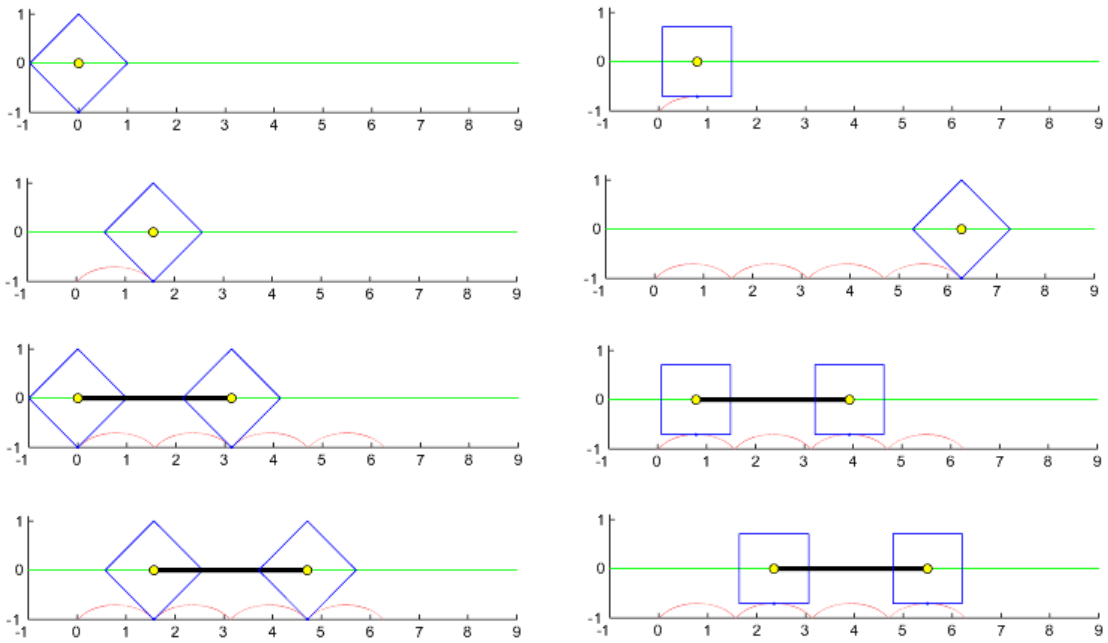
MATLAB-Skriptfile

```
% car01.m
% Simulation Autofahrt mit Raedern = Quadrat bzw. Kreis
% k=1..4 -> 1 Achse
% k=5..6 -> 2 Achsen

clear all
clc

eh=(0:0.02:2)*pi;
x0=cos(eh);
y0=sin(eh);
eh0=-0.3;
fig=figure;
hold on
axis equal;
axis([-1 9 -1 1.1]);
```

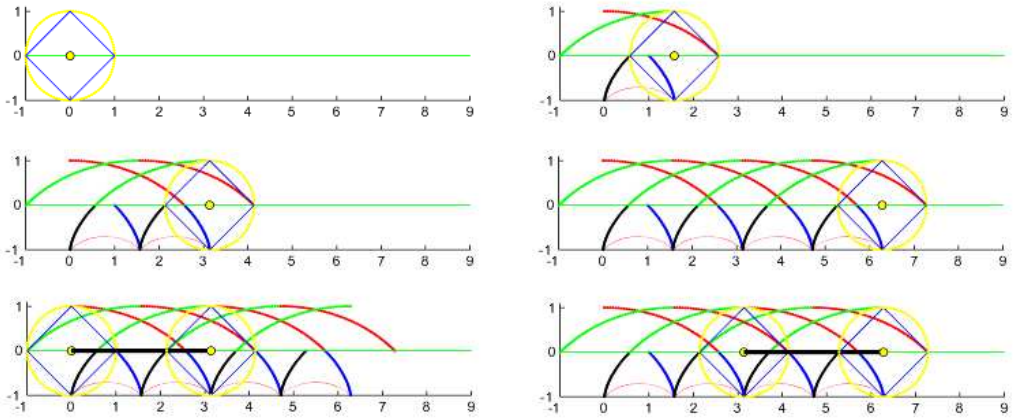
```
% Initialisierung
vier=plot(0,0);
vier2=plot(0,0);
kreise=vier;
center=vier;
point_tan=vier;
ax=vier;

kreise2=vier;
center2=vier;
point_tan2=vier;
i=4; j=5;
drawnow;

xx=0;
for k=1:6
 if k==5, xx=0; end;
 for eh0=-(0:0.01:0.5)*pi
   set(vier,'Visible','off')
   set(center,'Visible','off')
   set(point_tan,'Visible','off')
   % mit 1. gelben Kreis
   % set(kreise,'Visible','off')
   ox=abs(eh0)+xx;
   plot([-1 10],[0 0],'-g');
   eh_q=[0 (pi/2) pi (3*pi/2) (2*pi)]+eh0;
   xq=cos(eh_q)+ox;
   yq=sin(eh_q);
   dx=xq(j)-xq(i);
   dy=yq(j)-yq(i);
   m=dy/dx;
   x=ox;
   y=m*(x-xq(j))+yq(j);
   % Bewegungsablaeufe der 4 Ecken des Quadrats
   % plot(xq(1),yq(1),'.b');
   % plot(xq(2),yq(2),'.r');
   % plot(xq(3),yq(3),'.g');
   % plot(xq(4),yq(4),'.k');
   plot(x,y,'r');
   point_tan=line(x,y,'Marker','.');
   % mit 1. gelben Kreis
   % kreise=plot(x0+ox,y0,'.y');
   center=line(ox,0,'Marker','o','MarkerFaceColor','y','color','k');
   vier=line(xq,yq);
   % 2 Achsen
   if k>=5
```

```
      % set(kreise2,'Visible','off')
      set(vier2,'Visible','off')
      set(center2,'Visible','off')
      set(point_tan2,'Visible','off')
      set(ax,'Visible','off')
      xxx=[0 pi]+ox;
      yyy=[0 0];
      ax=plot(xxx,yyy,'-k','LineWidth',3);
      point_tan2=line(x+pi,y,'Marker','.');
      center2=line(ox+pi,0,'Marker','o','MarkerFaceColor','y','color','k');
      vier2=line(xq+pi,yq);
      % mit 2. gelben Kreis
      % kreise2=plot(x0+ox+pi,y0,'.y');
    end
    drawnow;
    % pause
    % pause(0.001)
    % close(fig);
  end
  xx=ox;
end
```

Fahrt mit einer Achse bzw. einem Auto auf quadratischen Rädern über die Huckelpiste



Fahrt mit einer Achse bzw. einem Auto auf quadratischen Rädern
Zuschaltung der kreisrunden Räder sowie Anzeige der Bahnkurven der Ecken des quadratischen Rads (Herausnehmen der Kommentare im Skriptfile `car01.m`)

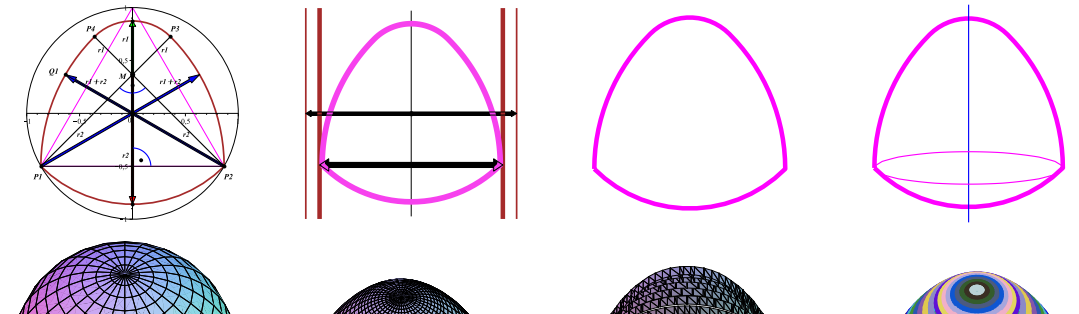


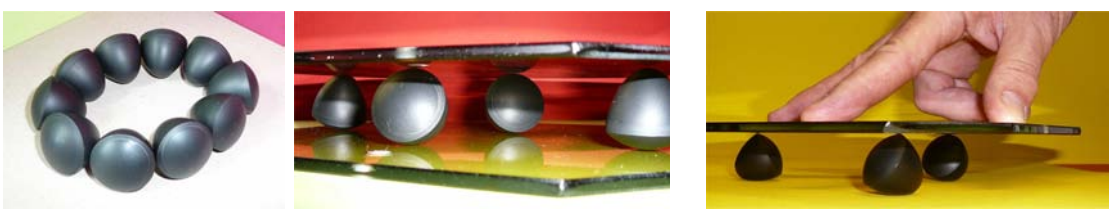
2.6.9 Körper gleicher Dicke

Verzeichnis: `solids_pseudokugeln`
Datei: rotation_dreieck_rund4.mws
Modelle: Kugel, Pseudokugel

Von der Kugel zum Wespennest

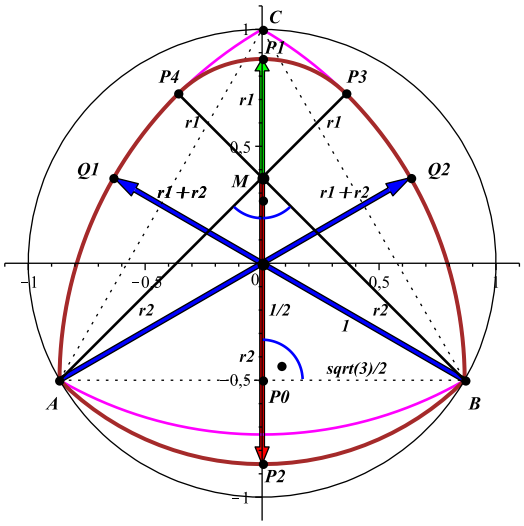
Der Körper entsteht durch Rotation mit dem Zweieck als Mantellinie.





Voraussetzung ist die Formel der Mantellinie $DZ(t) = (x(t), y(t))$, $0 \le t \le 2\pi$, des Zweiecks bei stückweiser Definition entsprechend der nebenstehenden Abbildung, als Verallgemeinerung der Randkurve $DR(t)$ des abgerundeten Dreiecks.

Das allgemeine Zweieck setzt sich aus den 4 Kurventeilen AB, BP_3, P_3P_4 und P_4A zusammen. θ ist der Winkel $\angle(A, M, P2)$ mit $\pi/6 \le \theta \le \pi/2$. Falls $\theta = \pi/4$, dann sind die Geraden AM und BM senkrecht zueinander.



Maple-Anweisungen zur Abbildung mit Zweieck

```
> rho:=1:
  theta:=Pi/4:    # spezielles Zweieck
  n:=120:         # Feinheit der Kurve, bei Dreieck reichen 3 Punkte
                  # durch 12 teilbar
  dtn:=2*Pi/n:

> # Dreiecke, abgerundet, Beschriftung
  P3:=dreieck_rund2(Pi);
  P4:=dreieck_rund2(3*Pi/2);
  Q1:=rho*[-r12*cos(Pi/6)+sqrt(3)/2,r12*sin(Pi/6)-1/2];
  Q2:=[-Q1[1],Q1[2]];
  P0:=rho*[0,-1/2]; P1:=M+rho*[0,r1]; P2:=M-rho*[0,r2];

> liste:=[seq(dreieck1(i*dtn),i=0..n)]:    # 3 Eckpunkte reichen
  lister:=[seq(dreieck_rund1(i*dtn),i=0..n)]:
  listerr:=[seq(dreieck_rund2(i*dtn),i=0..n)]:

> p1n:=plot(liste,color=black,axes=normal,thickness=1,linestyle=dot):
  p1rn:=plot(lister,color=magenta,axes=normal,thickness=2):
  p1rr:=plot(listerr,color=brown,axes=normal,thickness=3):
  kr:=implicitplot(x^2+y^2=rho^2,x=-rho..rho,y=-rho..rho,color=black):

  wr1:=plot({[A,P3],[B,P4]},color=black,thickness=2):
  wr2:=pointplot([A,B,C,P3,P4,Q1,Q2,P0,P1,P2,M-[0,0.1],[0.08,-0.44]],
                 color=black,symbol=solidcircle,symbolsize=16):
  wr3:=pointplot([nu,M],color=black,symbol=solidcircle,symbolsize=20):

  tt1:=textplot([[-0.9,-0.6,‘ A‘],[0.9,-0.6,‘ B‘],[0.05,1.05,‘ C‘],
        [0.07,-0.57,‘ P0‘]],font=[TIMES,BOLD,11]):
  tt2:=textplot([[0.4,0.8,‘ P3‘],[-0.4,0.8,‘ P4‘],[-0.1,0.35,‘ M‘],
        [-0.75,0.4,‘ Q1‘],[0.75,0.4,‘ Q2‘],[0.05,M[2]-r2-0.05,‘ P2‘],
        [0.05,M[2]+r1+0.05,‘ P1‘]],font=[TIMES,BOLD,11]):
  tt3:=textplot([[-0.5,-0.2,‘ r2‘],[0.5,-0.2,‘ r2‘],[-0.07,-0.4,‘ r2‘],
        [0.3,0.6,‘ r1‘],[-0.3,0.6,‘ r1‘],[-0.07,0.7,‘ r1‘],
        [0.35,0.3,‘ r1+r2‘],[-0.35,0.3,‘ r1+r2‘],[-0.35,0.3,‘ r1+r2‘],
```

```
        [0.4,-0.45,‘ sqrt(3)/2‘],[0.35,-0.27,‘ 1‘],[0.07,-0.2,‘ 1/2‘]],
        font=[TIMES,BOLD,10]):
pf1:=[arrow(M,P1,0.02,0.06,0.10,color=green),
      arrow(M,P2,0.02,0.06,0.06,color=red)]:
pf2:=[arrow(B,Q1,0.02,0.06,0.06,color=blue),
      arrow(A,Q2,0.02,0.06,0.06,color=blue)]:
kru1:=plot(M[2]-sqrt(0.03-x^2),x=-0.12..0.12,color=blue,thickness=2):
kru2:=plot(-0.5+sqrt(0.03-x^2),x=0..0.18,color=blue,thickness=2):

> display(p1n,p1rn,p1rr,kr,wr1,wr2,wr3,tt1,tt2,tt3,pf1,pf2,kru1,kru2,
      scaling=constrained,view=[-rho-0.1..rho+0.1,-rho-0.1..rho+0.1]);
```

3 Varianten der Erzeugung des Körpers (zu den 3 Wespennestern)

Dazu benötigen wir einige Hilfsfunktionen sowie die Hälfte der Mantellinie $DZ(t) = (x(t), y(t))$ des Zweiecks.

```
> # Variabler Mittelpunkt Mv, mittig von oben nach unten laufend
  # Mv:  C --> M --> M' --> P0,  theta=30..60..90
  Mv:=theta->piecewise(theta=Pi/2,-rho/2,
                       rho*(sqrt(3)/(2*tan(theta))-1/2));
  Mv(Pi/6); Mv(Pi/4); Mv(Pi/3); Mv(Pi/2);   # C, M, M', P0

  r2f:=theta->rho*sqrt(3)/(2*sin(theta));
  r1f:=theta->rho*sqrt(3)-r2f(theta);
  r12:=rho*sqrt(3);

> # Randkurve des Zweiecks
  # DZ(0)=DZ(2Pi)=A, DZ(Pi/2)=B, DZ(Pi)=P3, DZ(3Pi/2)=P4
  dreieck_rund3:=unapply(piecewise(
     phi<Pi/2,[0+r2f(theta)*cos(3*Pi/2+4/Pi*theta*(phi-Pi/4)),
               Mv(theta)+r2f(theta)*sin(3*Pi/2+4/Pi*theta*(phi-Pi/4))],
     phi<Pi,  [-rho*sqrt(3)/2+r12*cos((1-2*theta/Pi)*(phi-Pi/2)),
               -rho/2+r12*sin((1-2*theta/Pi)*(phi-Pi/2))],
     phi<3*Pi/2,[0+r1f(theta)*cos(Pi/2+4/Pi*theta*(phi-5*Pi/4)),
        Mv(theta)+r1f(theta)*sin(Pi/2+4/Pi*theta*(phi-5*Pi/4))],
       [ rho*sqrt(3)/2+r12*cos((1-2*theta/Pi)*(phi-3*Pi/2)+Pi/2+theta),
        -rho/2+r12*sin((1-2*theta/Pi)*(phi-3*Pi/2)+Pi/2+theta)]),phi);

> ha:=sqrt(3)/2;
  r2:=sqrt(2)*ha; r1:=(sqrt(2)-1)*r2; r12:=r1+r2;

  # Kontrolle der Punkte
  M:=rho*[0,(sqrt(3)-1)/2];
  A:=rho*[-sqrt(3)/2,-1/2];
  B:=rho*[sqrt(3)/2,-1/2];
  C:=rho*[0,1];
  P2:=dreieck_rund3(Pi/4);
  P3:=dreieck_rund2(Pi);
  P4:=dreieck_rund2(3*Pi/2);
  Q1:=rho*[-r12*cos(Pi/6)+sqrt(3)/2,r12*sin(Pi/6)-1/2];
  Q2:=[-Q1[1],Q1[2]];
  P0:=rho*[0,-1/2]; P1:=M+rho*[0,r1]; P2:=M-rho*[0,r2];
```

(1) Körper als "Schlauch"

Drehung der Mantellinie $DZ(t) = (x(t), y(t)),\ \pi/4 \le t \le 5\pi/4$, ("Hälfte" von $DZ(t)$) im 3D um die z-Achse bedeutet:
Rotationsachse ist z-Achse, also $(0, 0, y(t))$, $x(t)$ als Radiusfunktion.

```
> theta:=Pi/4;
  pl1:=tubeplot([0,0,dreieck_rund3(t)[2]],t=Pi/4..5*Pi/4,
                radius=dreieck_rund3(t)[1],tubepoints=40,
                view=[-1.3..1.3,-1.3..1.3,-1.3..1.3],axes=none):
  pl2:=pointplot3d({[-1.3,-1.3,0],[1.3,1.3,0]},color=white):  # Hilfe
  display(pl1,pl2);
```

(2) Körper aus "schmalen" Streifen zusammensetzen

```
> dw:=10:          # Anzahl der Streifen = 180/dw
  pkkh:=array(0..180/dw,[]):
  pkk:=array(0..180/dw,[]):
  In:=-1.5..1.5:
  for k from 0 by dw to 180 do
   phi:=Pi/4+k*Pi/180;
   rs:=dreieck_rund3(phi)[1]; zs:=dreieck_rund3(phi)[2];
   pkkh[k/dw]:=implicitplot3d(x^2+y^2=rs^2,
           x=-1..1,y=-1..1,z=zs-0.05..zs+0.05,grid=[30,30,2],
           style=patch,transparency=0.5,view=[In,In,In]):
   pkkhh:=display(seq(pkkh[l],l=0..k/dw),view=[In,In,In]);
   pl2a:=pointplot3d({[-1.5,-1.5,0],[1.5,1.5,0]},color=white):  # Hilfe
   pkk[k/dw]:=display(pkkhh,pl2a);
  end do:

> pkks:=[seq(pkk[k],k=0..180/dw)]:
  display(pkks,insequence=true);
```

(3) Körper aus "dicken" Kurven in horizontalen Ebenen übereinander

```
> dw:=5:           # Anzahl der Kurven = 180/dw
  pkkh:=array(0..180/dw,[]):
  pkk:=array(0..180/dw,[]):
  In:=-1.5..1.5:
  pkkh[0]:=pointplot3d([0,0,P2[2]],color=black,
                       symbol=solidcircle,symbolsize=16):
  pkk[0]:=pkkh[0]:
  for k from dw by dw to 180 do
   phi:=Pi/4+k*Pi/180;
   rs:=dreieck_rund3(phi)[1]; zs:=dreieck_rund3(phi)[2];
   pkkh[k/dw]:=spacecurve([rs*cos(t),rs*sin(t),zs],
              t=0..2*Pi,thickness=5,view=[In,In,In],
              color=COLOR(RGB,rand()/10^12,rand()/10^12,rand()/10^12)):
   pkkhh:=display(seq(pkkh[l],l=0..k/dw),view=[In,In,In]);
   pkk[k/dw]:=display(pkkhh,pl2a);      # pl2a Hilfe
  end do:

> pkks:=[seq(pkk[k],k=0..180/dw)]:
  display(pkks,insequence=true);
```

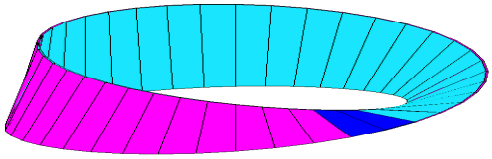
2.7 Ringe, Bänder und Scherenschnitte

2.7.1 Möbiusband und Riemen

Verzeichnis: `flaechen_3d\moebiusband`
Dateien: ringband1.m, mobiusband1,2.m
Modelle: Ringe, Bänder, Möbiusband, Riemen

Betrachten wir einen Streifen, der auf seiner Unter- und Oberseite verschieden gefärbt sein soll. Welche Form hat er nun, wenn man ihn mit verdrehten Enden zusammenklebt? Wir machen also einen einfachen Dreh, d.h. eine halbe Umdrehung.

Wir erhalten das **Möbiusband**.



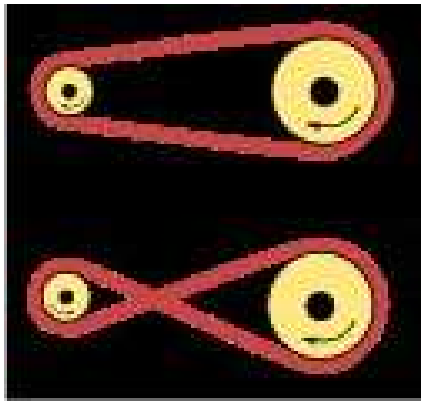
Dieses Band fasziniert nicht nur Mathematiker.
Die wundersame Schleife wurde bereits 1858 vom Leipziger Mathematiker und Astronomen **August Ferdinand Möbius** (1790 - 1868) entdeckt.
Die verblüffenden Eigenschaften der Schleife haben nicht nur Künstler inspiriert. Ingenieure montieren Förderbänder und Antriebsriemen als Möbiusband, damit sich beide Seiten des Bandes gleichmäßig abnutzen. Sie nutzen die Tatsache, dass es bei der Schleife weder oben und unten noch innen und außen gibt.

Ich habe das Möbiusband schon in meiner frühen Kindheit kennengelernt, ohne damals überhaupt irgendetwas über das Band zu wissen und damit seine Bedeutung erkennen zu können. Aber es war der Alltag, das Leben und die Arbeit auf dem Dorf, die den Umgang mit Bändern und damit auch mit Möbiusbandern erfordeten.
In der einzelbäuerlichen Landwirtschaft meiner Eltern gab es zahlreiche Maschinen und Geräte, die sogenannte Riemenräderwerke (Riementriebe, Riemenscheibentriebe) hatten. Dazu gehörten insbesondere die stationäre Dreschmaschine in der Scheune mit Strohpresse und Sackheber, aber auch die Futterschneidemaschine, Schüttelmaschine, Rübenschnitzelmaschine, Schrotmühle, Mähbinder oder Kreissäge. Überall gab es einen Motor mit Antriebsrad, Riemenräder (umlaufende Räder, Scheiben) und Treibriemen. Die Riemen waren Flachriemen, meist aus Leder, manchmal aus Kunststoff. Die Räder waren zumeist fest. Riemenspanner gab es nicht. Die Laufseite

der Riemen musste wegen der Griffigkeit regelmäßig mit Fett oder Talg eingerieben werden.
Die Lederflachriemen waren meistens an den Enden mit speziellen Klammern (Riemenzungen und Stahlstift) miteinander verbunden. Das klammerförmige Verbindungsstück war ohne weiteres lösbar (trennbare Riemenverbindung). Damit konnte man den Riemen eventuell kürzen oder nach einem Dreh wieder zusammenbauen. Die lösbare Verbindung hatte den Vorteil, dass man die Riemen auch an unzugänglichen Stellen ohne Ausbau der Antriebsscheiben auflegen konnte.



Wenn sich bei einem Riementrieb beide Räder im gleichen Sinn bewegen, spricht man von einem offenen Riementrieb. Bei umgekehrtem Sinn bzw. Rotationsumkehr wird der Riemen in Form einer 8 aufgezogen und das heißt gekreuzter Riementrieb. Es gibt noch weitere Riementriebe mit unterschiedlichen Riementypen, z.B. geschränkte Riemenführung mit Rundriemen oder Keilriemenantrieb.

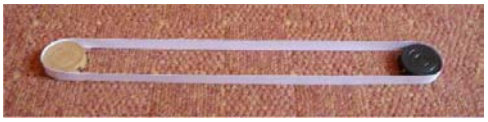


Bei der Arbeit mit den oben genannten Maschinen hat von Zeit zu Zeit die Riemenspannung nachgelassen. Eine Möglichkeit, um bei längeren Flachriemen (lange Transmissionen) die Spannung wieder zu erhöhen, bestand im Lösen der Riemenverbindung und Drehen der Enden. So machte ein einfacher Dreh Sinn, also die Simulation des Möbiusbandes, oder auch ein Doppeldreh. Natürlich war das abhängig davon, ob der Riemenantrieb offen oder gekreuzt war.

Schauen wir einige Situationen etwas genauer an.

(1) Band ohne Dreh

Man kann es für den offenen und gekreuzten Riementrieb verwenden.




(2) Band mit einem Dreh, Möbiusband
Man kann es für den offenen Riementrieb verwenden. Beim gekreuzten Riementrieb würde eine Bandkante an der Bandfläche schleifen.

(3) Band mit Doppeldreh
Man kann es für den offenen Riementrieb verwenden. Beim gekreuzten Riementrieb würden beide Bandkanten aneinander reiben.

Wenn man ein Band an den Enden mit Doppeldreh (1 volle Umdrehung) zusammenklebt, dann entsteht i. Allg. automatisch eine von den zwei Figuren:
Die Ziffer 8 bzw. ein Kreisel.

Das sind beides Schleifen, wo es oben und unten bzw. innen und außen gibt. An der Farbunterscheidung der Seiten wird es deutlich.

Wer einmal auf der Achterbahn gefahren ist, hat vielleicht noch den Kreisel in Erinnerung. Wegen seiner mathematischen Verwandschaft zur 8 gehört er als Element auch zur Achterbahn. Es gibt den Kreisel auch mancher Orten als Kunstwerk.

Wenn man den Kreisel etwas breit drückt, erkennt man eine nach oben gewölbte 8.

Die 8 ist sozusagen die duale Figur zum Kreisel. Von der nach oben gewölbte 8 stülpen wir eine Seite um, und schon entfaltet sich die 8.

Die mathematische Beschreibung der 8 ist nicht einfach das Zusammenfügen von zwei Kreisen, sondern eine Lemniskate, ein Sonderfall der Cassinischen Kurve.
Ihr implizite Darstellung ist

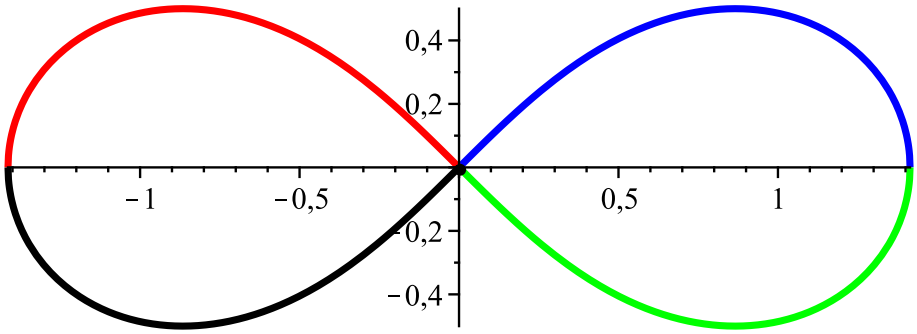
$$(x^2+y^2)^2 \;=\; 2a^2(x^2-y^2), \quad a>0,$$

und verweist auf ihre symmetrischen Eigenschaften.
In Polarkoordinaten notieren wir die Formeln

$$r \;=\; a\sqrt{2\cos(2\varphi)}, \quad \varphi \in [-\tfrac{\pi}{4},\tfrac{\pi}{4}] \cup [\tfrac{3\pi}{4},\tfrac{5\pi}{4}].$$

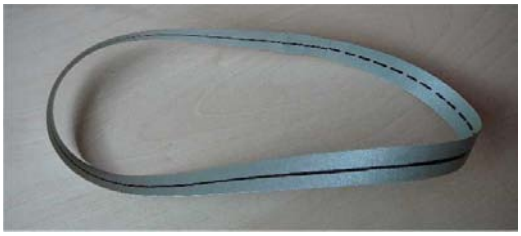
Der Koordinatenursprung ist ein Doppelpunkt und zugleich ein Wendepunkt.
Die ebene Ausdehnung der Kurve liegt im Rechteck $[-a\sqrt{2}, a\sqrt{2}] \times [-a/2, a/2]$.
Die vier Kurventeile beschreiben wir im kartesischen Koordinatensystem mit Maple.

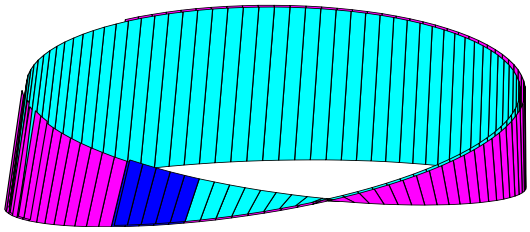
```
> a:=1;
  r:=a*sqrt(2);
  setoptions(thickness=3):
  p1:=plot([a*sqrt(t*(1+t)),a*sqrt(t*(1-t)),t=0..1],color=blue):
  p2:=plot([-a*sqrt(t*(1+t)),a*sqrt(t*(1-t)),t=0..1],color=red):
  p3:=plot([a*sqrt(-t*(1-t)),-a*sqrt(-t*(1+t)),t=-1..0],color=green):
  p4:=plot([-a*sqrt(-t*(1-t)),-a*sqrt(-t*(1+t)),t=-1..0],color=black):
  p5:=pointplot([0,0],symbol=solidcircle,symbolsize=16,color=black):

> display(p1,p2,p3,p4,p5,scaling=constrained);
```

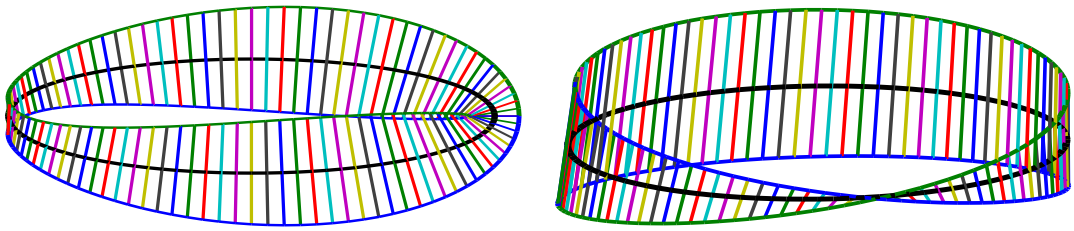


Simulation des Möbiusbandes in MATLAB

Für gute Ansichten des Bandes sowie spezielle Effekte ist der Aufwand etwas größer und es sind einige Tricks anzuwenden.







```
% Darstellung des Moebiusschen Bandes mit Leitkreis
% Papierstreifen, der an den Enden um 180 gedreht und zusammengeklebt wird
%
% Geometrisch: Durch A=(r,0,-la) und B=(r,0,la) ist eine vertikale Strecke
% der Laenge 2*la gegeben.
% Diese Strecke wird nun so um die z-Achse rotiert, dass sich ihr Mittelpunkt
% M gleichmaessig auf dem Kreis (Leitkreis) mit Radius r=5 und Zentrum (0,0,0)
% in der Grundrissebene bewegt. Gleichzeitig wird die Strecke um M in einer
% Ebene, die die z-Achse enthaelt, gedreht, und zwar soll sich die Strecke
% bei Rueckkehr zum Ausgangspunkt um 180 Grad gedreht haben.
%
% Darstellung des Moebiusschen Bandes mit Leitellipse
%
% Geometrisch: Durch A=(a,0,-la) und B=(a,0,la) ist eine vertikale Strecke
% der Laenge 2*la gegeben.
% Diese Strecke wird nun so um die z-Achse rotiert, dass sich ihr Mittelpunkt
% M gleichmaessig auf der Ellipse (Leitellipse) mit den Halbachsen
% a=10 und b=1 und Zentrum (0,0,0) in der Grundrissebene bewegt ...
%
% Die Punkte des Bandes sind vektoriell
% Variante 1
%  P=((r+la*cos(phi2))cos(phi),(r+la*cos(phi2))sin(phi),la*sin(phi2))
%     phi2=phi/2
%
% Variante 2 (z.T. guenstiger)
%  P=(r*cos(phi)+la*cos(phi2),r*sin(phi)+la*cos(phi2),la*sin(phi2))
%
% Andere Baender: phi2 konstant bzw. phi2=phi, 2*phi, ...

% Moebius-Band
% Schnitt an der schwarzen Kurve (1/2-Hoehe)
a=10;
ra=1.5*a;
b=1;
rb=2*b;
la=1;           % teste la=2,3,...
la1=2*la;
n=101;
```

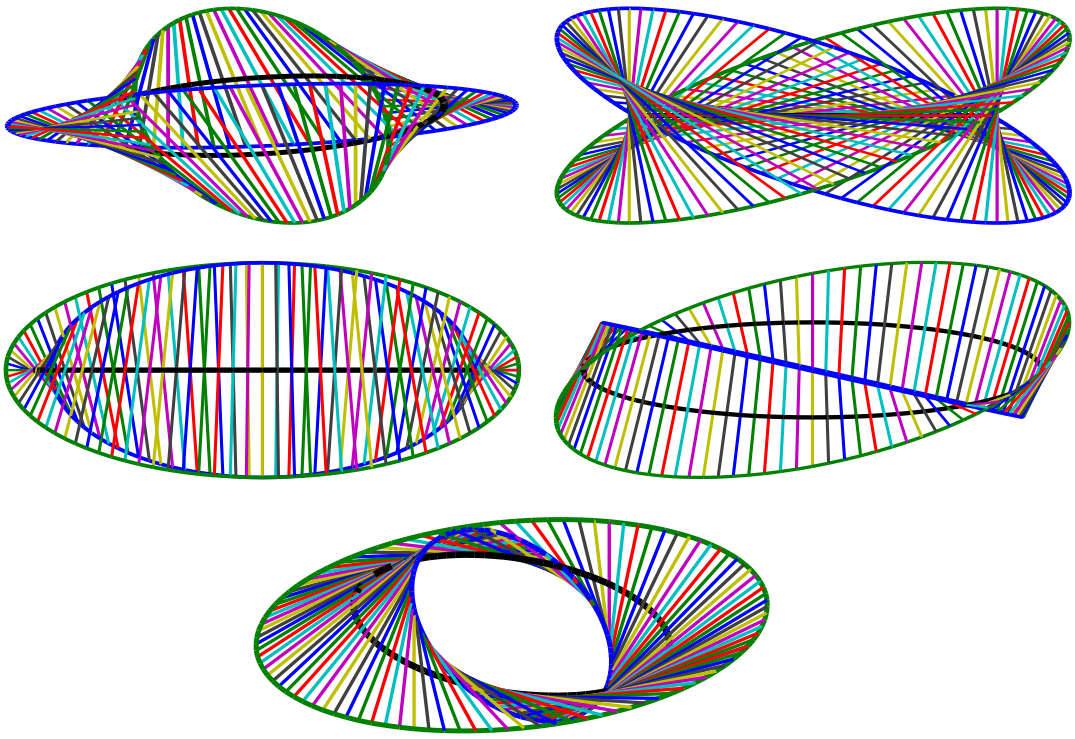
```
phi=linspace(0,2*pi,n);
xm=a*cos(phi);
ym=b*sin(phi);
zm=zeros(size(xm));

figure(1)
rox=a+la*cos(phi/2); xo=rox.*cos(phi);
roy=b+la*cos(phi/2); yo=roy.*sin(phi);
zo=la*sin(phi/2);
rux=a-la*cos(phi/2); xu=rux.*cos(phi);
ruy=b-la*cos(phi/2); yu=ruy.*sin(phi);
zu=-zo;
a1=a+0.2; b1=b+0.03;
rox1=a1+la*cos(phi/2); xo1=rox1.*cos(phi);
roy1=b1+la*cos(phi/2); yo1=roy1.*sin(phi);
zo1=la*sin(phi/2);
rux1=a1-la*cos(phi/2); xu1=rux1.*cos(phi);
ruy1=b1-la*cos(phi/2); yu1=ruy1.*sin(phi);
zu1=-zo1;
plot3([xu;xo],[yu;yo],[zu;zo],'LineWidth',2);
view(0,75)   % view(45,45)
axis off
axis([-ra ra -rb rb -la1 la1]);
hold on
plot3(xu,yu,zu,xo,yo,zo,'LineWidth',3);
plot3(xm,ym,zm,'k','LineWidth',4);
print -dpsc 'moebius15.ps'
print -djpeg 'moebius15.jpg'
hold off
pause

% oder guenstiger
figure(2)
xo=a*cos(phi)+la*cos(phi/2);
yo=b*sin(phi)+la*cos(phi/2);
zo=la*sin(phi/2);
xu=a*cos(phi)-la*cos(phi/2);
yu=b*sin(phi)-la*cos(phi/2);
zu=-zo;
xo1=a1*cos(phi)+la*cos(phi/2);
yo1=b1*sin(phi)+la*cos(phi/2);
zo1=la*sin(phi/2);
xu1=a1*cos(phi)-la*cos(phi/2);
yu1=b1*sin(phi)-la*cos(phi/2);
zu1=-zo1;
```

```
plot3([xu1;xo1],[yu1;yo1],[zu1;zo1],'LineWidth',2);
view(-5,130)
axis off
axis([-ra ra -rb rb -la1 la1]);
hold on
plot3(xu1,yu1,zu1,xo1,yo1,zo1,'LineWidth',3);
plot3(xm,ym,zm,'k','LineWidth',4);
print -dpsc 'moebius15m.ps'
print -djpeg 'moebius15m.jpg'
hold off
pause

%------------------------------------------------------------------------
figure(6)
X41=zeros(4,n-1); Y41=X41; Z41=X41;
for i=1:n-1
 X41(1,i)=xu1(i);   Y41(1,i)=yu1(i);   Z41(1,i)=zu1(i);
 X41(2,i)=xo1(i);   Y41(2,i)=yo1(i);   Z41(2,i)=zo1(i);
 X41(3,i)=xo1(i+1); Y41(3,i)=yo1(i+1); Z41(3,i)=zo1(i+1);
 X41(4,i)=xu1(i+1); Y41(4,i)=yu1(i+1); Z41(4,i)=zu1(i+1);
end
fill3(X4(:,20:n-1),Y4(:,20:n-1),Z4(:,20:n-1),...
      'c','FaceAlpha',1);
hold on
fill3(X41(:,[1:14,20:n-17]),Y41(:,[1:14,20:n-17]),Z41(:,[1:14,20:n-17]),...
      'm','FaceAlpha',1);
hh=ones(4)*0.05;
fill3(X41(:,3:6)-hh,Y41(:,3:6),Z41(:,3:6)+0.5*hh,'m','FaceAlpha',1); % Trick
fill3(X4(:,15:19),Y4(:,15:19),Z4(:,15:19),'b','FaceAlpha',1);
fill3(X41(:,15:19),Y41(:,15:19),Z41(:,15:19),'b','FaceAlpha',1);
view(-5,130)
axis off
axis([-ra ra -rb rb -la1 la1]);
print -dpsc 'moebius19.ps'
print -djpeg 'moebius19.jpg'
pause
```

Ein Band mit voller Drehung kann analog implementiert werden.
Ansichten des Bandes können zwar schön, aber auch durchaus verwirrend sein.

```
% Band mit voller Drehung (Doppeldreh)
n=101;
phi=linspace(0,2*pi,n);
phi2=phi;
xm=a*cos(phi); ym=b*sin(phi); zm=zeros(size(xm));
```

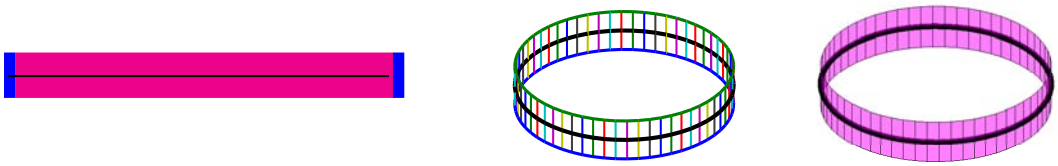
```
a1=a+0.2; b1=b+0.03;
figure(8)
xo=a*cos(phi)+la*cos(phi2); yo=b*sin(phi)+la*cos(phi2);
zo=la*sin(phi2);
xu=a*cos(phi)-la*cos(phi2); yu=b*sin(phi)-la*cos(phi2);
zu=-zo;
xo1=a1*cos(phi)+la*cos(phi2); yo1=b1*sin(phi)+la*cos(phi2);
zo1=la*sin(phi2);
xu1=a1*cos(phi)-la*cos(phi2); yu1=b1*sin(phi)-la*cos(phi2);
zu1=-zo1;

plot3([xu1;xo1],[yu1;yo1],[zu1;zo1],'LineWidth',2);
view(-5,130)
axis off
axis([-ra ra -rb rb -la1 la1]);
hold on
plot3(xu1,yu1,zu1,xo1,yo1,zo1,'LineWidth',3);
plot3(xm,ym,zm,'k','LineWidth',4);
print -dpsc 'moebius21.ps'
print -djpeg 'moebius21.jpg'
hold off
pause
```



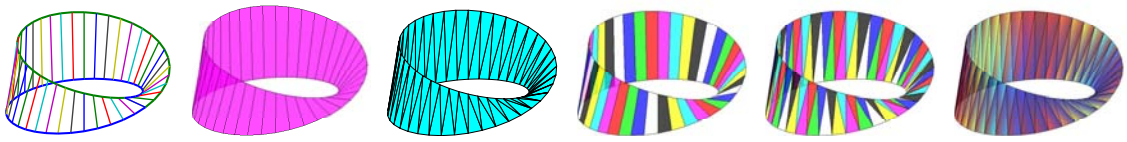
2.7.2 Ring und Möbiusband

Verzeichnis: `flaechen_3d\moebiusband`
Dateien: ringband1.m, mobiusband1,2.m
Modelle: Ringe, Bänder, Torus

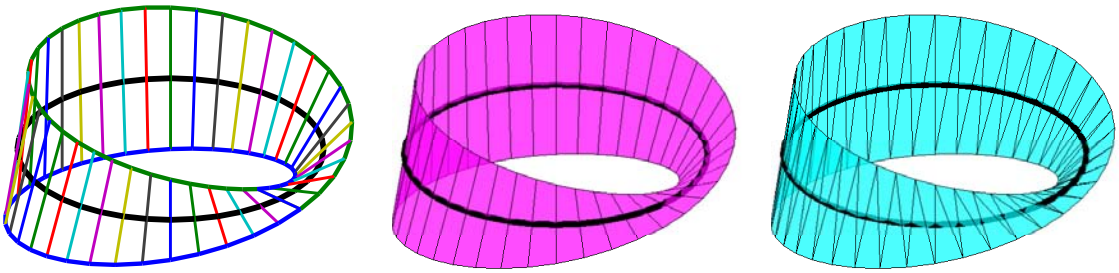
Ringband mit Verkleben der beiden Streifenenden



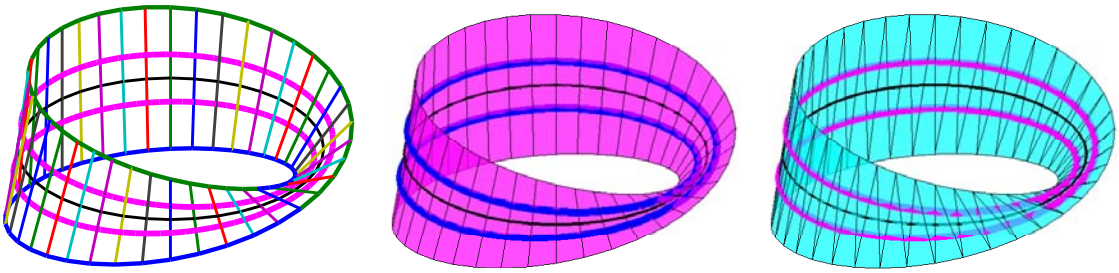
Möbius-Band in Variationen mit Verkleben der Streifenenden mit Dreh



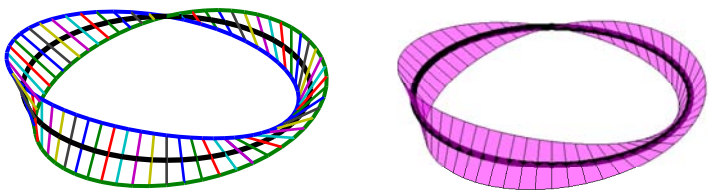
mit mittlerer Schnittkurve



mit Schnittkurve bei 1/3



Verkleben der beiden Streifenenden mit Doppeldreh (volle Umdrehung)



2.7.3 Scherenschnitte oder Experimente mit Papier

Verzeichnis: `flaechen_3d\moebiusband`
Dateien: moebiusband1,2.m
Modelle: Ringe, Bänder

Das Basteln und Kleben von Ringen und Möbiusbändern sowie das weitere Zusammenkleben zu doppelten Ringen und Bändern bildet die Grundlage für unterschiedliche Schnittmuster.

Erstaunliches geschieht, wenn man einem Möbiusband mit einer Schere zu Leibe rückt. Was passiert zum Beispiel, wenn ein Band entlang der Mittellinie in Längsrichtung zerschnitten wird? Zerfällt es in zwei Hälften? Mitnichten: Es entsteht ein neues Band doppelter Länge, das doppelt verdreht ist, statt nur um eine halbe Drehung wie beim Möbiusband.

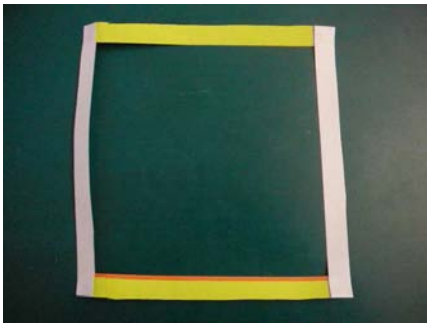
Noch verrückter ist das Ergebnis beim Dritteln des Streifens entlang der Längsachse: Jetzt zerfällt das Band tatsächlich - aber in zwei ineinander verschlungene Schleifen, eines davon ist wieder ein Möbiusband. Das zweite Band ist doppelt verdreht.

Also entstehen bei Längsschnitten von Bändern und ihren Verbindungen z.B. wieder Bänder, Handschellen, Fotorahmen oder verschlungene Herzen.

Die Ergebnisse sollen hier demonstriert werden.

Ringband, Möbius-Band, über Kreuz geklebte Bänder

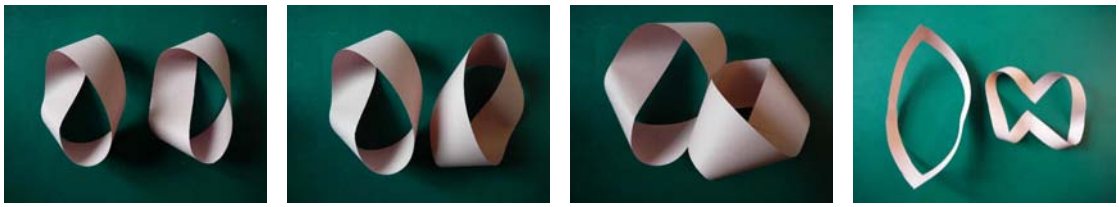
Längsschnitte bei einem Ring und bei zwei über Kreuz zusammengeklebten Ringen



Möbiusband entlang der Längsachse halbieren, führt zu einem doppelt verdrehten Band

Möbiusband entlang der Längsachse dritteln, führt zu “Handschellen“

2 Möbiusbänder (nebeneinander), eins drehen, dann über Kreuz zusammenkleben und entlang der Längsachsen halbieren, zerfallen in 2 Bänder (“Schiff“ und “Fast-8“



2 Möbiusbänder (gespiegelt nebeneinander) über Kreuz zusammenkleben und entlang der Längsachsen halbieren, zerfallen **nicht** und führen zu verschlungenen Herzen

Möbiusband und Ring über Kreuz zusammenkleben und entlang der Längsachsen halbieren, führen auf einen Fotorahmen

2.8 Mathematikum

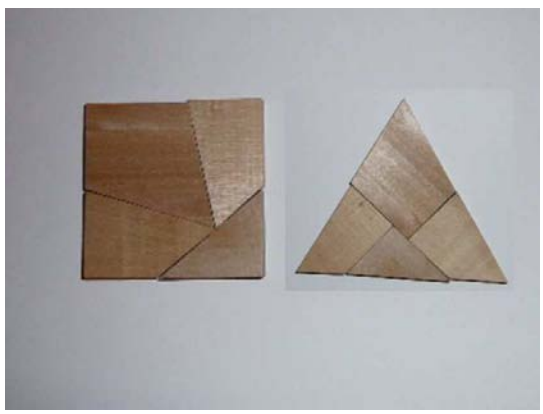
Das Mitmach-Museum rund um die Mathematik in Gießen, das erste mathematische Mitmachmuseum der Welt, ist einen Besuch wert.

```
Mathematikum
www.mathematikum.de
www.mathematikum-unterwegs.de
Liebigstrasse 8
35390 Giessen
Tel. 0641 / 969797-0
```

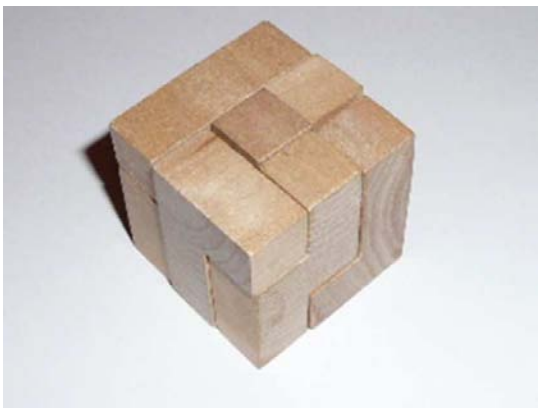
Es bietet natürlich Mathematik in vielen Facetten, aber auch die Lange Nacht der Mathematik, Vorträge, Experimente, Kindervorlesungen, **Beutelspachers Sofa** und viele andere Veranstaltungen. Dazu gibt es Shop und OnlineShop. Eine Bonderheit sind Wanderausstellungen. Einige Exponate dieser Ausstellung "Mathematik zum Anfassen" sind auf der Internetseite vorgestellt.

- Da waren's nur noch 14 ...
- Wer kommt als erster ins Ziel?
- Der Drehspiegel
- Das Quadratpuzzle
- Das Tensegrity

Mit alldem wird die Mathematik leicht und anschaulich gemacht. Das Mathematikum in Gießen möchte die Mathematik einer möglichst großen Allgemeinheit zugänglich machen. Werfen wir einen Blick auf kleine Puzzles des Mathematikums.




HOLZPUZZLE
Der Zauberwürfel
The magic cube
Le cube magique
7 Teile / 7 pièces / 7 pieces




DAS WABENPUZZLE
mathematikum


HOLZPUZZLE
Das Kreuz
The cross
Le croix


HOLZPUZZLE
Wooden puzzle / Puzzle en bois
Die Kugelpyramide
The ball pyramide
Le pyramide des boules

Kapitel 3

Schlussbemerkungen

3.1 Geschichte der Mathematik einmal anders

Mit der Gründung der Hochschule für Elektrotechnik 1953 wurde die akademische Ingenieurausbildung in Ilmenau fortgesetzt. Damit verbunden war auch die Wiederbelebung des studentischen Lebens und seiner Traditionen. Zu den Höhepunkten für die Studentenschaft zählt das Bergfest nach der Hälfte des Studiums.

Die folgende "mathematische Abhandlung" findet man in der Bergfestvorlesung des Jahres 1963, veröffentlicht in den **Bergfesthalbheiten** *bh8* vom 29.6.1963.

Einführung in die Geschichte der Mathematik

Es war noch im transzendenten Zeitalter. Aus indefinitiven Gründen lebten die Algebraiker und die Analytiker in Feindschaft miteinander. Alle Friedenskonferenzen waren durch die Konstanz der Repräsentanten der United States of Algebraikum gescheitert: Diese riefen bei solchen Gelegenheiten immer: "My-Ny, My-Ny,...". Die Feindschaft wäre bis ins Unendliche gegangen, wenn nicht die folgenden kommensurablen Ereignisse eingetreten wären.
Die Algebraiker, es war im letzten akademischen Viertel vor Euklids Geburt, hatten (diesmal aus sehr definitiven Gründen) eine endliche Reihe von schönen Analysierinnen in ihre abstrakten Gefilde entführt. Die Aufregung war in sämtlichen Quadranten der Analytiker absolut genommen sehr groß. Ihre Majoranten stellten einen Plan der Befreiung auf, den sie das Erlanger Programm nannten, und der letzte Polyhistor

begann das heute noch bekannte Werk “Der Raub der Analysierinnen“ zu schreiben. Der freie Vektor, der bislang in der n-ten Dimension ein mathematisches Insulanerdasein geführt hatte, erfuhr von allen diesen Geschehnissen erst in der Abendausgabe der “Riemannschen Blätter“. Er konnte darob nicht schlafen, so heftig alternierte er. Es hatte gerade die Weltzeitstunde geschlagen, da erschien ihm die heilige Analysis, die Schutzpatronin der Analytiker und sprach: “0h, freier Vektor, Du allein kannst die Lösung finden. Errichte Dich und konvergiere gegen Algebraikum! Rette die Ehre der Adjunktfrauen! Gauß schütze Dich!“
Als der Morgen graute, rüstete sich der Vektor. Er cramerte auf, zählte seine Absolutbeträge nach, gürtete sich mit einem Sarrus, ergriff Bogen und Bolzanos und schloß sein System ab.
Auf der Ordinate Analysiums, der Weierstraße, wo die Majoranten die Minoranten zu beruhigen suchten, wurde der freie Vektor mit wissenschaftlich exaktem Jubel begrüßt und gefeiert, als er von der nächtlichen Induktion sprach. Alle kamen zu dem Entschluß, in Analogie zu ihrer Schutzpatronin, dann und nur dann könne der Disput gewonnen werden, wenn er als Einzelelement in den Kampf zöge. Sie begleiteten ihn rechts und von links bis zur oberen Schranke, und dann zog der freie Vektor allein auf der Zahlengeraden weiter.
Unterwegs traf er einen alten Mann, der im Sande projizierte. Der freie Vektor teilte ihm harmonisch mit, was er für seine Zukunft voraussetze, erhielt jedoch die grobe Schlußfolgerung: “Störe meine Kreise nicht!“ Das war der weise Apollonios. Der rüstige Wanderer ließ sich nunmehr von keiner Seite in keiner Weise tangieren. Versuchte einer, ihn von der Zahlengeraden abzulenken, dachte er bei sich: “Der ist bestimmt divergent!“ und “Thales, Thales, du mußt wandern ...“. Vor sich hinpfeifend, konvergierte er vondannen.
Als der Einheitskreis seine letzten Polaren über die Ebene sandte, erreichte unser Vektor einen Kurvenwald. Gleich am Rande hockte unter einer Wurzel ein Euler. Kaum war der Vektor in dessen Sektor, rief der Euler sein schauriges “i-i-i-i-...“. Der freie Vektor dachte: “Gauß vergib ihm, denn er weiß nicht, was er tut!“
Im geometrischen Mittel des Waldes lauerte eine Lemniskate auf ihrer Sprungstelle. Unser Held war weniger als ein Epsilon heran, da begann das cassinische Ungeheuer, das per Definition gar nicht hierher gehörte, so wild zu oszillieren, auf daß er mannigfaltig beschränkt ward. In höchster Not bemerkte er, daß er in einem Wald zweiter Ordnung war. Er riß einen Hyperbelast ab und schlug solange zu, bis von dem Feind nur noch ein singulärer Punkt verblieb. Nach diesem Abenteuer war es für den freien Vektor trivial, die Maxima, Minima, Wendepunkte und Nullstellen dieses ansonsten so schwer zu diskutierenden Waldes zu überwinden. Der Einheitskreis schob sich segmentweise über den Horizont, als der freie Vektor den Grenzübergang zum Algebraikum erreichte, das in einem zusammenhängenden offenen Gebiet lag. In monoton wachsendem Tempo integrierte der Analytiker über die Flächen und erreichte bald den limes inferior. Zwischen diesem und dem limes superior lag ein großes Intervall. Da er in seiner Dienstzeit beim ersten akademischen Semester zu Fuß gestanden hatte, war es für ihn trivial, eine Schachtelung anzusetzen und so den Torus von Algebralkum zu erreichen. Die Algebraiker hatten sich hinter dem limes superior unendlich

klein gemacht, obwohl das nicht exakt war. Jetzt aber hielten sie die Peripherie fest. Die einzelnen Gruppen verteidigten die Häufungspunkte. Am Häufungspunkt der Häufungspunkte von Häufungspunkten stand die Kleinsche Vierergruppe mit der zyklischen und alternierenden Reihe.
Im ersten Augenblick war der freie Vektor komplex, so sehr fixierten sie ihn. Dann rief er: "Euklid, ich habs! Es gibt nur ein Mittel zum Siege" Durch Multiplikation mit sich selbst erzeugte er seine n-dimensionale Mannigfaltigkeit. Mit dieser Potenz gerüstet, trieb er die Gruppen und dann auch ihre Untergruppen in Fluchtgruppen. Nur den Rest hatte er nicht genügend abgeschätzt, denn er hatte es mit der alternierenden Gruppe besonders schwer. Diese wollte jeder Umlegung widerstehen. Selbstkritisch wie er war, dachte er: "Pi" und wandte einen Kunstgriff an. Geschickt wie Archimedes schleuderte er seine Bolzanos, auf daß die einzelnen Glieder absolut genommen und damit alle konvergent wurden. Ihre Restglieder flohen darauf längs des Radius zum Zentrum. Man sah leicht, daß alles einfach war. Auf dem Kreisring standen die Algebraiker, allen voran die Primzahlen. Sie machten vor dem siegreichen freien Vektor ihr algebraisches Kompliment und übergaben ihm das Signum von Algebraikum. Schließlich führten sie die nunmehr befreiten Analysierinnen in der natürlichen Folge vor, allen voran die Matrix und die Determinante. Dem freien Vektor, der so lange von der insularen Ideologie befallen war, gefielen die beiden par excellenz. Jedoch schüchtern, wie er war, permutierte er erst ein Weile. Dann aber kam er zu den Voraussetzungen. Es war zuerst notwendig, die Ideale ihrer Körper zu bestimmen. Nach der hinreichenden Behauptung, sie genügten einer Ungleichung, kam er zum Beweis und schließlich zur Definition: "Matrix, daraus folgt: vorne und hinten rund - unberechenbar! Determinante, daraus folgt: berechenbar, da vorne und hinten glatt! Auf Grund dieser Tatsache folgt: ich will mich mit Euch verbinden. Dieses Tripel wird eine glückliche Komposition zur Erzeugung eines Ringes mittels Unterkörpern nach der Methode Dedekind".
Bei den Algebraikern nennt man so eine Verbindung Distributivehe. So geschah es, daß der freie Vektor ein gebundener wurde.
Mit isomorphem Talent verband er Analysium mit Algebraikum. Dann trat er zum algebraischen Glauben über, nach dem Grundsatz: Algebraikum ist eine Messung wert, und baute den Unterworfenen den Tempel des Abel. Der freie Vektor, Gauß sei Dank, nun ein gebundener, wurde konstant zum Prinzeps Mathematikorum ausgerufen. Und wenn er nicht expliziert, so existiert er noch heute in unserer Anschauung.

Bergfesttext von Dipl.-Ing. Arndt Albrecht
Student der 4. Matrikel von 1956 bis 1962
an der Hochschule für Elektrotechnik Ilmenau

3.2 Zusammenfassung

Die erfolgreiche Anwendung von Methoden und Verfahren für wissenschaftliche aber auch unterhaltsame Aufgabenstellungen hängt nicht nur von ausgereiften Softwaretools ab. In der ersten Phase der Lösung ist eine breite fachliche Grundbildung für Problemanalyse, für Möglichkeiten der eventuellen Umformung einer Aufgabe in eine andere adäquate Darstellung sowie für die Auswahl von entsprechenden Lösungsmethoden notwendig. Dazu gehören natürlich die geschickte Umsetzung einer Anwendungssituation in ein passendes (mathematisches) Modell sowie die kreative Entwicklung und sachgemäße Beurteilung und Begründung mathematischer Begriffe und Verfahren. Dieser Kenntnisstand kann zu eleganten und effizienten Algorithmen, einer höheren Qualität der Verfahren und der darauf aufbauenden Computerprogrammen führen.
Andererseits verfügen auch die Softwarewerkzeuge über zahlreiche tutorielle Elemente und sie können in vielfältiger Weise zu tiefergreifenden Überlegungen und neuen Ideen führen. Dabei bieten sich insbesondere der Einsatz von Computeralgebrasystemen (CAS) wie Maple, Matlab oder *Mathematica* sowie von Programmierumgebungen mit Pascal, Delphi oder C an.
Die CAS erfahren eine ständige Weiterentwicklung und werden immer größer und leistungsfähiger. Programme und Arbeitsblätter, die unter älteren Versionen entstanden und gelaufen sind, sollte man bezüglich der Möglichkeiten und Veränderungen in neuen Versionen stets kritisch begutachten und eventuell anpassen.

Auswahl einiger Softwareaspekte als methodisch-didaktische Hilfsmittel:

- Multiple-window Darstellungen, Desk-Top-Publishing Systeme,
- Computergraphik, algebraische, geometrische und graphische Symbolsysteme,
- Darstellung von Formeln und Tabellen,
- Datenspeicherung und -manipulation,
- Wiederholbarkeit, Reflexion, Interaktion, Simulation.

An konkreten Beispielen zeigt sich, was ein Computer schon und noch nicht leisten kann. Der wirklich kreative Anteil am Problemlösungsprozess bleibt beim Menschen. Der Phantasie sind aber keine Grenzen gesetzt. Natürlich ist es zumeist nicht einfach, komplizierte Sachverhalte methodisch geschickt und didaktisch wirksam für Präsentationen vor einem Hörerkreis aufzubereiten.
Denkt man bei Computerunteranwendungen mehr an den akademischen und wissenschaftlichen Bereich, zum Beispiel an adaptive und selbstkorrigierende Verfahren, Intervallarithmetik, Konzept der hohen und optimalen Genauigkeit, wissenschaftliches Rechnen mit Ergebnisverifikation oder an paralleles und verteiltes Rechnen, so sind hier schon neue und erfolgversprechende Wege gegangen worden.
So erhält man mit Systemen des wissenschaftlichen Rechnens neben einem Ergebnis auch sehr nützliche Informationen über seine Bewertung und Brauchbarkeit.
Solche und noch nicht vorstellbare Entwicklungen sind ernst zu nehmen, um die komplexen und wachsenden Probleme zu meistern.

Literaturverzeichnis

[1] Neundorf, W.: *Numerische Mathematik*. Vorlesungen, Übungen, Algorithmen und Programme. Shaker Verlag, Aachen 2002.

[2] Werner, W.: *Mathematik lernen mit Maple*. Ein Lehr- und Arbeitsbuch für das Grundstudium. Band 1, 2. dpunkt-Verlag, Heidelberg 1996, 1998.

[3] Neundorf, W.: *Kondition eines Problems und angepasste Lösungsmethoden*. Preprint No M 9/95 April 1995 IfMath TU Ilmenau.

[4] Neundorf, W.: *Gewöhnliche Differentialgleichungen - Beispiele, Modelle, Verfahren, Software*. Preprint No M 11/98 Mai 1998 IfMath TU Ilmenau.

[5] Neundorf, W.: *Programming in Maple V Release 5*. Extended Basics. Preprint No M 07/99 Februar 1999 IfMath TU Ilmenau.

[6] Neundorf, W.; Walther, B.: *Grafik, Animation und Ausgabeformate in Maple V Release 5*. Preprint No M 12/00 Juli 2000 IfMath TU Ilmenau.

[7] Neundorf, W.: *Lösungsmethoden mit Maple*. Betrachtung von Fehlern und Kondition sowie Darstellungsmöglichkeiten. Preprint No M 08/03 April 2003 IfMath TU Ilmenau.

[8] Neundorf, W.: *Spezielle Aspekte zu CAS Maple und Matlab*. Rechengenauigkeit, Listen, Felder, Faktorisierungen, Dateiarbeit, TP → Maple, Maple → MATLAB mit Beispielen. Preprint No M 10/03 Juni 2003 IfMath TU Ilmenau.

[9] Neundorf, W.: *π und e*. Preprint No M 06/04 März 2004 IfMath TU Ilmenau.

[10] Neundorf, W.: *Kondition eines Problems sowie Gleitpunktarithmetik in den CAS Maple, MATLAB und in höheren Programmiersprachen*. Preprint No M 16/07 November 2007 IfMath TU Ilmenau.

[11] Neundorf, W.: *Fourier-Reihen, π und Cotangens*. Preprint No M 36/09 November 2009 IfMath TU Ilmenau.

[12] Neundorf, W.: *Die komplexen Wurzeln aus 1*. Preprint No M 01/10 Januar 2010 IfMath TU Ilmenau.

[13] Hromkovic, Juraj: *Sieben Wunder der Informatik*. Eine Reise an die Grenze des Machbaren mit Aufgaben und Lösungen. B.G. Teubner Verlag, Wiesbaden 2006.

[14] Kortenkamp, Ulrich, TU Berlin; Richter-Gebert, Jürgen, TU München: *CAS, Dynamische Geometrie Software Cinderella*, interaktive Geometrie mit Cinderella, *Visage*: Visualisierung von Algorithmen,

MATHEON DFG Forschungszentrum, Dynamische Geometrie an der Schnittstelle von Schule und Hochschule

[15] Rovenski, Vladimir: *Geometry of Curves and Surfaces with Maple.* Birkhäuser Verlag, Boston Basel Berlin 2000.

[16] Bryant, John; Sangwin, Chris: *How Round Is Your Circle?* Where Engineering and Mathematics Meet. Princeton University Press, 2008.

[17] Beutelspacher, Albrecht; Wagner, Marcus: *Wie man durch eine Postkarte steigt ... und andere spannende mathematische Experimente.* Herder-Verlag, Freiburg 2008.

[18] Aigner, Martin; Ziegler, Günter M.: *Das BUCH der Beweise.* Springer-Verlag, Heidelberg Berlin 2002.

[19] Zimmermann, B.: *Zur Heuristik in der Geschichte der Mathematik und im Mathematikunterricht.* Vortrag zur DMV-Tagung, Jena 1996.

[20] Lakatos, Imre: *Beweise und Widerlegungen.* Vieweg-Verlag, Wiesbaden Braunschweig 1979.

[21] Simonyi, Karoly.: *Kulturgeschichte der Physik* von den Anfängen bis 1990. Verlag Harri Deutsch, Thun Frankfurt/Main 1995.

[22] Bednarczuk, J. u.a. (Hrsg.): *Matematyka.* Czasopismo dla nauczycieli. Nr.12/2000, Verlag WSiP, Wrocław 2000.

[23] Juszkiewicz, A.P. (Red.): *Historia matematyki.* Teil I, II, III. PWN, Warszawa 1975-77, Übersetzung aus dem Russ.

[24] Steeb, Willi-Hans: *Chaos and Fractals.* Algorithms and Computations. BI Wissenschaftsverlag, Mannheim 1992.

[25] O'Connor, J.J.; Robertson, E.F.: *History Topics.*
http://www-history.mcs.st-and.ac.uk/Indexes/Hist_Topics_alph.html

[26] O'Connor, J.J.; Robertson, E.F.: *Mathematical games and recreations.*
http://www-history.mcs.st-and.ac.uk/HistTopics/Mathematical_games.html

[27] Glatz, P.; Moldenhauer, W.: *Zum 150. Todestag des Princeps mathematicorum.* Thillm, Bad Berka 2005.

[28] Fröhlich, I. u.a. (Hrsg.): *Paxis der Mathematik.* Heft 9, Juni 2006/48. Jg. Aulis Verlag Deubner, Köln und Leipzig 2006.

[29] Stengel, H.; Müller, R.F.: *Die Wortspielwiese.* Der Kinderbuchverlag, Berlin 1982.

[30] Havas, H.; Mündemann, B.-M.: *Power Training für den Kopf.* Buch und Zeit Verlagsgesellschaft mbH, Köln 2008.

Anschrift:

Dr. rer. nat. habil. Werner Neundorf
Technische Universität Ilmenau, Institut für Mathematik
PF 10 05 65
D - 98684 Ilmenau

E-mail : werner.neundorf@tu-ilmenau.de
Homepage : http://www.tu-ilmenau.de/fakmn/neundorf.html