MWS zu Kapitel 1

MWS 1

Die komplexen Zahlen

Wichtige MAPLE-Befehle dieses Kapitels:

interface, z. B. interface(imaginaryunit=i)
alias, macro
Re, Im, abs, conjugate, polar, argument
textplot, complexplot, coordplot, conformal, display
plottools-Paket: arc, arrow, disk, line, transform
radnormal
cat
map, op

1.1 Historisches

Von der ‚Erfindung' der komplexen Zahlen

```
> restart: with(plots):
  interface(imaginaryunit=i):
```

Durch die interface-Anweisung imaginaryunit $= i$ wird das standardmäßig vordefinierte Symbol I für die imaginäre Einheit $\sqrt{-1}$ deaktiviert und durch i ersetzt.

Folgendes Beispiel stammt aus Geronimo CARDANOs *Ars Magna*. Gesucht sind (reelle?) Zahlen x_1 und x_2 mit $x_1 + x_2 = 10$ und $x_1 \cdot x_2 = 40$. Nach dem VIETAschen Wurzelsatz lassen sich x_1 und x_2 als die Lösungen einer quadratischen Gleichung auffassen. Die dazu nötigen Umformungs- und Lösungsschritte führen wir — zur Einübung — mit einfachen Maple-Anweisungen durch:

```
> eq1 := x[1]+x[2]=10; eq2 := x[1]*x[2]=40;
```

$$eq1 := x_1 + x_2 = 10\,, \quad eq2 := x_1\,x_2 = 40$$

Wir lösen die erste Gleichung nach x_2 auf und setzen das Ergebnis in die zweite Gleichung ein:

```
> solve(eq1,{x[2]});
```

$$\{x_2 = -x_1 + 10\}$$

```
> eq3 := subs(%,eq2);
```

$$eq3 := x_1 (-x_1 + 10) = 40$$

Substituieren wir x_1 durch z, so ist folgende quadratische Gleichung zu lösen:

```
> eq4 := subs(x[1]=z,eq3);
```

$$eq4 := z (-z + 10) = 40$$

```
> expand(-eq4);
```

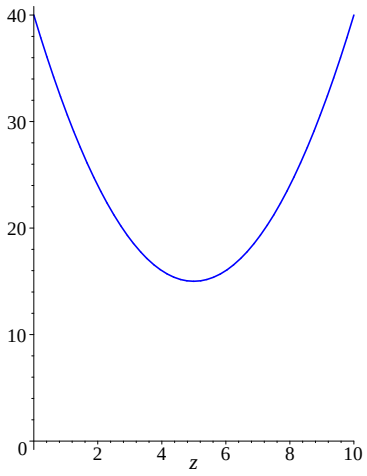
$$z^2 - 10\,z = -40$$

```
> eq5 := lhs(%)-rhs(%) = 0;
```

$$eq5 := z^2 - 10\,z + 40 = 0$$

Durch die linke Seite dieser Gleichung ist — für reelle z — eine Parabel definiert, deren Nullstellen gesucht sind. Wir veranschaulichen den Funktionsverlauf:

```
> plot(lhs(eq5),z=0..10,color=blue,view=[0..10,0..40],thickness=2);
```



Spätestens damit ist klar, daß die Parabel keine reelle Nullstelle besitzt. Mit Hilfe des Maple-Kommandos completesquare aus dem student-Paket läßt sich dies natürlich auch durch ganz einfache algebraische Umformung einsehen:

```
> with(student):  eq6 := completesquare(eq5,z);
```

$$eq6 := (z-5)^2 + 15 = 0$$

Maple kennt die komplexen Zahlen und kann uns somit die Lösungen explizit angeben:

```
> solve(eq6,z);
```

$$5 + i\sqrt{15}, \quad 5 - i\sqrt{15}$$

Auf genau diese Form ist CARDANO durch formales Lösen der quadratischen Gleichung gekommen, wobei er Schwierigkeiten hatte, dem Symbol $i := \sqrt{-1}$ eine konkrete Bedeutung zu geben. In der Bezeichnung *imaginäre Einheit* lebt dies fort.

Durchs Imaginäre zurück ins Reelle

```
> restart:
  interface(imaginaryunit=i): macro(I=i):
```

Wie oben vereinbaren wir mittels der interface-Anweisung für $\sqrt{-1}$ die von EULER eingeführte Bezeichnung i. Die Neudefinition von I durch den macro-Befehl `I=i` bewirkt dann, daß ein eingegebenes I als imaginäre Einheit interpretiert und stets als i ausgegeben wird.

Beachten: i kann dann z. B. *nicht als Laufvariable* benutzt werden! (Das gilt natürlich auch schon auf Seite 15.)

Das folgende Beispiel einer kubischen Gleichung geht auf Rafael BOMBELLI zurück.

```
> eq := z^3+p*z+q;
  p := -15; q := -4;
```

$$eq := z^3 + p\,z + q, \quad p := -15, \quad q := -4$$

Die folgenden Rechenschritte orientieren sich an unseren Überlegungen aus Abschnitt 1.1 des Textes. Zur Vereinfachung eines Ausdrucks mit Wurzeltermen muß dabei statt *simplify* oder *normal* das Maple-Kommando `radnormal` verwendet werden.

```
> A := -q/2+sqrt((q/2)^2+(p/3)^3):
  u := radnormal(A^(1/3));
  v := -p/3/u;
  omega := (-1+I*sqrt(3))/2;
```

$$u := 2 + i, \quad v := 2 - i, \quad \omega := -1/2 + 1/2\,i\sqrt{3}$$

Mit den *Cardanischen Formeln* ergibt sich:

```
> z1 = u+v;
  u*omega+v*omega^2:  z2 = simplify(%);
  u*omega^2+v*omega:  z3 = simplify(%);
```

$$z1 = 4, \quad z2 = -2 - \sqrt{3}, \quad z3 = -2 + \sqrt{3}$$

Maple findet natürlich die gleichen Lösungen auch direkt:

```
> solve(eq,z);
```

$$4, \quad -2 + \sqrt{3}, \quad -2 - \sqrt{3}$$

1.2 Definition und Modelle komplexer Zahlen

Arithmetische Einführung der komplexen Zahlen

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

Die komplexen Zahlen als Zahlenpaare

Für das Rechnen mit Zahlenpaaren benutzen wir als *Addition* die standardmäßig vorhandene komponentenweise Addition (Vektoraddition).

```
> [x[1],y[1]] + [x[2],y[2]];
```

$$[x_2 + x_1,\ y_2 + y_1]$$

Die *Multiplikation* definieren wir mittels eines *„neutralen Operators“*, d. h. einer *Prozedur* folgenden Typs:

```
> `&.` := proc(u,v)                    # Multiplikation
    eval([u[1]*v[1]-u[2]*v[2],u[1]*v[2]+u[2]*v[1]])
  end:
```

Mit dieser Prozedur kann die Multiplikation auf zwei verschiedene Arten ausgeführt werden:

```
> &. ( [x[1],y[1]],[x[2],y[2]] );
```

$$[x_1\,x_2 - y_1\,y_2,\ x_1\,y_2 + y_1\,x_2]$$

```
> [x[1],y[1]] &. [x[2],y[2]];
```

$$[x_1\,x_2 - y_1\,y_2,\ x_1\,y_2 + y_1\,x_2]$$

Aufgrund der *Regeln für neutrale Operatoren* bindet die Operation „&.“ stärker als die Addition. Im Zweifelsfalle setze man passende Klammern!

```
> alias(i=[0,1]):     # i = imaginäre Einheit
```

Wir bestätigen die Beziehung $i^2 = -1$:

```
> i &. i;
```

$$[-1, 0]$$

Maple kennt natürlich die üblichen Rechenregeln. Wir verzichten jedoch darauf, diese alle zu verifizieren. Beispielhaft überprüfen wir nur die *Existenz eines inversen Elementes* für $[x, y] \neq 0$:

```
> [x,y] &. [u,v] = [1,0];
```

$$[x\,u - y\,v, x\,v + y\,u] = [1, 0]$$

```
> lhs(%)-rhs(%): eq := convert(%,set);
```

$$eq := \{-1 + x\,u - y\,v, x\,v + y\,u\}$$

In der folgenden Maple-Anweisung interpretiert *„solve“* die beiden Terme aus *„eq“* als homogene Gleichungen und löst diese nach den Unbekannten u, v auf.

```
> solve(eq,{u,v});
```

$$\left\{u = \frac{x}{x^2 + y^2}, \quad v = -\frac{y}{x^2 + y^2}\right\}$$

Die komplexen Zahlen als Unterring der 2×2-Matrizen

Jetzt benutzen wir als Operationen $+$ und $*$ die übliche Matrixaddition bzw. -multiplikation. Mittels folgender Zuordnung bilden wir geordnete Paare reeller Zahlen in 2×2-Matrizen ab:

```
> Phi := z -> matrix(2,2,[z[1],-z[2],z[2],z[1]]);
```

$$\Phi := z \to \mathrm{matrix}\big(2, 2, [z_1, -z_2, z_2, z_1]\big)$$

```
> 'Phi([x,y])' =  evalm(Phi([x,y]));
```

$$\Phi([x, y]) = \begin{bmatrix} x & -y \\ y & x \end{bmatrix}$$

Φ ist verträglich mit der Addition und der Multiplikation. Wir überprüfen dies mit Maple, indem wir die rechte Seite des folgenden Ausdrucks *eq* von der linken Seite abziehen, das Ergebnis in eine Menge konvertieren und mit der *if-Abfrage* feststellen, ob deren Elemente gleich 0 sind.

```
> eq := Phi([x[1],y[1]] + [x[2],y[2]])
      = Phi([x[1],y[1]])+ Phi([x[2],y[2]]):
  evalm(lhs(eq)-rhs(eq)): convert(%,set):
  if %={0} then eq fi;
```

$$\begin{bmatrix} x_2 + x_1 & -y_2 - y_1 \\ y_2 + y_1 & x_2 + x_1 \end{bmatrix} = \begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{bmatrix} + \begin{bmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{bmatrix}$$

```
> eq := Phi([x[1],y[1]] &. [x[2],y[2]])
      = Phi([x[1],y[1]]) &* Phi([x[2],y[2]]):
  evalm(lhs(eq)-rhs(eq)): convert(%,set):
  if %={0} then eq fi;
```

$$\begin{bmatrix} x_1\,x_2 - y_1\,y_2 & -x_1\,y_2 - y_1\,x_2 \\ x_1\,y_2 + y_1\,x_2 & x_1\,x_2 - y_1\,y_2 \end{bmatrix} = \begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{bmatrix} \,\&* \begin{bmatrix} x_2 & -y_2 \\ y_2 & x_2 \end{bmatrix}$$

Die Abbildung Φ ist zudem injektiv und somit ein Isomorphismus auf die Bildmenge der geordneten Paare. Dies ist eine Teilmenge der 2×2-Matrizen, die bezüglich $+$ und $*$ die Körperaxiome erfüllt. Nullelement ist in diesem Falle die Nullmatrix, Einselement die Einheitsmatrix. Das inverse Element zu $[x,\, y] \neq 0$ liest man aus den folgenden Überlegungen ab:

```
> Phi([x,y]) &* inverse(Phi([x,y])): % = simplify(evalm(%));
```

$$\begin{bmatrix} x & -y \\ y & x \end{bmatrix} \,\&* \begin{bmatrix} \dfrac{x}{x^2+y^2} & \dfrac{y}{x^2+y^2} \\ -\dfrac{y}{x^2+y^2} & \dfrac{x}{x^2+y^2} \end{bmatrix} = \mathrm{matrix}\big([[1,\,0],\,i]\big)$$

Maple liefert hier als Ergebnis die Einheitsmatrix in einer sehr verschlüsselten Form. Diese wird dargestellt als Liste der beiden Zeilenvektoren $[1,\,0]$ und $i = [0,\,1]$.

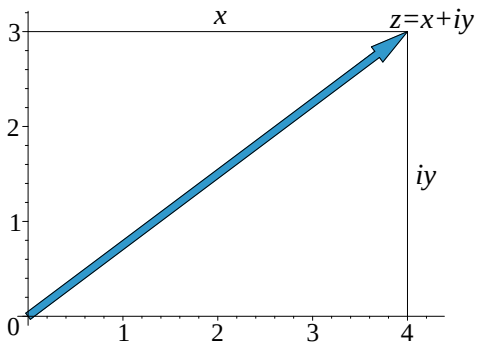
```
> alias(j=Phi([0,1])):      # j = imaginäre Einheit im Unterring
  j, evalm(j &* j);         # j^2 = -1
```

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Geometrische Einführung der komplexen Zahlen

```
> restart: with(plots): with(plottools):
  x := 4: y := 3: z := [x,y]:
  Pfeil  := arrow([0,0],z,0.08,0.2,0.08,color=cyan):
  hLinie := line([0,y],z,color=black): # horizontale Linie
  vLinie := line([x,0],z,color=black): # vertikale Linie
  Texte  := [x+0.3,y+0.15,"z=x+iy"],[x+0.2,y/2,"iy"],[x/2,y+0.15," x"]:
  tplot  := textplot([Texte]):         # Beschriftung
  display({Pfeil,hLinie,vLinie,tplot},tickmarks=[4,3],
          scaling=constrained, title="Die GAUSSsche Zahlenebene");
```

Die Gaußsche Zahlenebene



Hier tritt zum ersten Mal eine etwas komplexere Abbildung auf. Diese setzt sich zusammen aus vier sogenannten *Graphikstrukturen*, bei deren Benennung wir uns um möglichst suggestive Namen bemüht haben. Sie heißen *Pfeil*, *hLinie*, *vLinie* sowie *tplot*. Wir werden uns diese gleich im einzelnen etwas genauer anschauen. Zuvor sollten Sie jedoch den Maple-Befehl

```
> with(plottools);
```

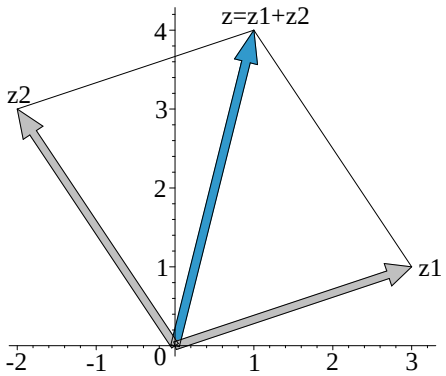
[*arc*, *arrow*, *circle*, *cone*, *cuboid*, *curve*, *cutin*, *cutout*, *cylinder*, *disk*, ...]

ausführen. Dieses Programm-Paket wurde oben (vor der Erzeugung der Abbildung) geladen. Man kann damit einfache Graphikstrukturen, wie z. B. *arc* (Kreisbogen), *arrow* (Pfeil), *disk* (Kreisscheibe), *line* (Strecke), erzeugen. Mit Befehlen wie *z. B. translate* (Verschieben), *scale* (Skalieren), *rotate* (Drehen) und *transform* können Graphikstrukturen transformiert werden. Nach Anklicken eines Befehls aus dem Plottools-Paket kann man sich mit *Help* darüber genauer informieren. Mit arrow($[x1, y1]$, $[x2, y2]$, $B1$, $B2$, V) erzeugt man einen Pfeil der Länge L von $[x1, y1]$ nach $[x2, y2]$. $B1$ ist die Pfeilbreite, $B2$ die Breite der Pfeilspitze und V das Verhältnis der Länge der Pfeilspitze zur Gesamtlänge L. line($[x1, y1]$, $[x2, y2]$) ergibt die Strecke von $[x1, y1]$ nach $[x2, y2]$. Mit der Option *color* lassen sich die einzelnen Graphik-Objekte wie üblich färben. Die Beschriftung innerhalb der Graphik erfolgt mit dem Kommando *textplot* aus dem plots-Paket. An drei Stellen soll hier ein Text ausgegeben werden. Die Textangabe erfolgt jeweils in der Form $[x, y, \text{"text"}]$, wobei *text* sowohl horizontal als auch vertikal bezüglich des Punktes $[x, y]$ zentriert wird. Die Folge *Texte* oben enthält drei solcher Textangaben, die in der Form [Texte] als Textliste durch textplot ausgegeben werden. Die Gesamtausgabe unserer Graphik-Bausteine erfolgt mittels display. Auf einfache Art kann man dabei die Graphik durch die Option title noch mit einer Überschrift versehen.

Die Addition komplexer Zahlen wird definiert als gewöhnliche Vektoraddition:

```
> x1 := 3: y1 := 1: x2 := -2: y2 := 3:
  z1 := [x1,y1]: z2 := [x2,y2]: z := z1+z2:
  Pfeil1 := arrow([0,0],z1,0.1,0.3,0.1,color=gray):
  Pfeil2 := arrow([0,0],z2,0.1,0.3,0.1,color=gray):
  Pfeil3 := arrow([0,0],z,0.11,0.3,0.1,color=cyan):
  Linien := plot([[z1,z],[z2,z]],color=black):
  Texte  := [x1+0.2,y1," z1"],[x2,y2+0.2," z2"],
            [op(z+[0.2,0.2]),"z=z1+z2"]:
  tplot  := textplot([Texte]):
  display({Pfeil1,Pfeil2,Pfeil3,Linien,tplot},scaling=constrained,
          title="Addition komplexer Zahlen");
```

Addition komplexer Zahlen



Natürlich läßt sich die Graphik noch verbessern; doch auf Einzelheiten dazu gehen wir erst etwas später ein.

Wie die Multiplikation geometrisch eingeführt werden kann, wird im Kontext von Abschnitt 1.4 verständlich. Der Nachweis der Körperaxiome erfolgt dann durch elementargeometrische Überlegungen.

1.3 Elementare Operationen und Regeln

Symbolisches Rechnen mit komplexen Zahlen

```
> restart: interface(imaginaryunit=i): macro(I=i):
  I; I^2;
```

$$i, \quad -1$$

```
> for k from 1 to 2 do z[k] := x[k]+I*y[k] od:
```

Summe, Produkt, Quotient (Inverse) komplexer Zahlen

```
> 'z[1]+z[2]' = evalc(z[1]+z[2]);
```

$$z_1 + z_2 = x_1 + x_2 + i\,(y_1 + y_2)$$

```
> 'z[1]*z[2]' = evalc(z[1]*z[2]);
```

$$z_1\,z_2 = x_1\,x_2 - y_1\,y_2 + i\,(x_1\,y_2 + y_1\,x_2)$$

```
> 'z[1]/z[2]' = evalc(z[1]/z[2]);
```

$$\frac{z_1}{z_2} = \frac{x_1\,x_2}{x_2^2+y_2^2} + \frac{y_1\,y_2}{x_2^2+y_2^2} + i\left(\frac{y_1\,x_2}{x_2^2+y_2^2} - \frac{x_1\,y_2}{x_2^2+y_2^2}\right)$$

```
> z := x + I*y:  '1/z' = evalc(1/z);
```

$$\frac{1}{z} = \frac{x}{x^2+y^2} + \frac{-i\,y}{x^2+y^2}$$

Real- und Imaginärteil

```
> 'Re(z)' = evalc(Re(z)); 'Im(z)' = evalc(Im(z));
```

$$\Re\mathfrak{e}(z) = x\,, \quad \Im\mathfrak{m}(z) = y$$

Durch Auswerten mittels der logischen Funktion evalb erhalten wir:

```
> eq := 'Re(z)' = '(z+conjugate(z))/2': evalc(%):
  if evalb(%) then eval(eq,1) fi;
```

$$\Re\mathfrak{e}(z) = \frac{1}{2}\,z + \frac{1}{2}\,\overline{z}$$

```
> eq := 'Im(z)' = (1/(2*I))*'(z-conjugate(z))': evalc(%):
  if evalb(%) then eval(eq,1) fi;
```

$$\Im\mathfrak{m}(z) = \frac{-1}{2}\,i\,(z - \overline{z})$$

Konjugation und Betrag

```
> 'conjugate(z)' = evalc(conjugate(z));
```

$$\overline{z} = x - i\,y$$

```
> 'abs(z)' = evalc(abs(z));
```

$$|z| = \sqrt{x^2+y^2}$$

Überprüfen einiger Rechenregeln

```
> eq := 'conjugate(z[1]+z[2])' = 'conjugate(z[1])+conjugate(z[2])':
  evalc(%): if evalb(%) then eval(eq,1) fi;
```

$$\overline{z_1+z_2} = \overline{z_1}+\overline{z_2}$$

```
> eq := 'conjugate(z[1]*z[2])' = 'conjugate(z[1])*conjugate(z[2])':
  evalc(%): if evalb(%) then eval(eq,1) fi;
```

$$\overline{z_1\,z_2} = \overline{z_1}\,\overline{z_2}$$

```
> eq := 'z*conjugate(z)' = 'abs(z)^2': evalc(%):
  if evalb(%) then eval(eq,1) fi;
```

$$z\,\overline{z} = |z|^2$$

Beispiele

```
> restart: interface(imaginaryunit=i): macro(I=i):
  u := 2-5*I; v := 4+I;
```

$$u := 2-5\,i, \quad v := 4+i$$

Summe, Produkt, Quotient komplexer Zahlen

```
> u+v; u-v; u*v; u/v;
```

$$6-4\,i, \quad -2-6\,i, \quad 13-18\,i, \quad \frac{3}{17}-\frac{22}{17}\,i$$

Eine 4-stellige Näherung für das vorangehende Resultat ergibt sich dann so:

```
> evalf(u/v,4);
```

$$.1765-1.294\,i$$

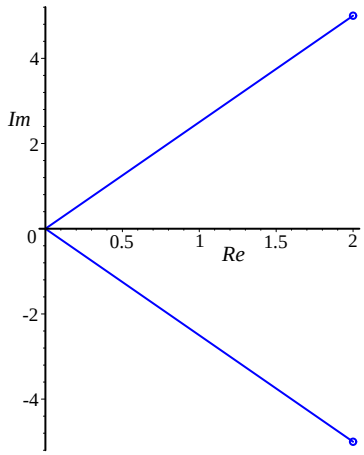
Real- und Imaginärteil, Konjugation und Betrag

```
> Re(u); Im(v); conjugate(u); abs(u);
```

$$2, \quad 1, \quad 2+5\,i, \quad \sqrt{29}$$

Komplexe Zahlen zeichnen

```
> with(plots):
  Linien := complexplot([u,0,conjugate(u)],thickness=2,color=blue):
  Punkte := complexplot([u,conjugate(u)],style=point,symbol=circle,
                        symbolsize=20,color=blue):
  display(Linien,Punkte,labels=[Re,Im],thickness=2);
```



Wir lernen hier den Maple-Befehl complexplot kennen. Analog zu *plot* kann man durch complexplot($[z1, z2, z3]$) eine Liste $[z1, z2, z3]$ komplexer Zahlen in der angegebenen Reihenfolge geradlinig verbinden. Durch die Option *style=point* werden die aufgelisteten Zahlen nur als Punkte ausgegeben. Mit der ab Maple 6 existierenden Option *symbolsize* kann deren Größe verändert werden.

1.4 Argument, geometrische Veranschaulichung

Eulersche Formel

```
> restart:
  interface(imaginaryunit=i): macro(I=i):
  exp(x+I*y) = evalc(exp(x+I*y));
```

$$e^{x+i\,y} = e^{x}\cos(y) + i\,e^{x}\sin(y)$$

```
> exp(I*x) = evalc(exp(I*x));
  Re(exp(I*x)) = evalc(Re(exp(I*x)));
  Im(exp(I*x)) = evalc(Im(exp(I*x)));
```

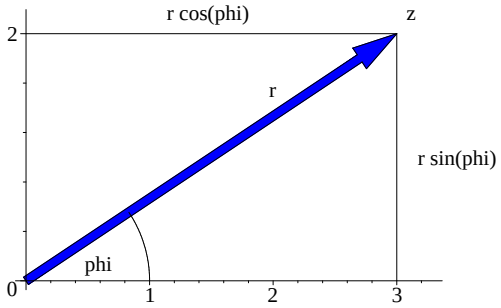
$$e^{i\,x} = \cos(x) + i\sin(x)\,, \quad \Re\mathfrak{e}\,(e^{i\,x}) = \cos(x)\,, \quad \Im\mathfrak{m}\,(e^{i\,x}) = \sin(x)$$

Polardarstellung komplexer Zahlen

Wie bei Vektoren im $\mathbb{R}^2$ kann man komplexe Zahlen mittels der Polardarstellung durch ihren Betrag r und ihr Argument φ beschreiben.

```
> restart: interface(imaginaryunit=i): macro(I=i):
  with(plots): with(plottools):
  x := 3: y := 2: z := x+y*I: phi := argument(z):
  Pfeil  := arrow([0,0],[x,y],0.07,0.2,0.1,color=blue):
  hLinie := line([0,y],[x,y],color=black):
  vLinie := line([x,0],[x,y],color=black):
  Arcus  := arc([0,0],1,0..phi,color=black,thickness=2): # Kreisbogen
  Texte  := [2/3*x,2/3*y+0.2,"r"],
            [0.6*cos(phi/2),0.5*sin(phi/2)," phi"],[x+0.1,y+0.15," z"],
            [x/2,y+0.15,"r cos(phi)"],[x+0.3,y/2,"r sin(phi)"]:
  tplot  := textplot([Texte]):
  display({Pfeil,hLinie,vLinie,Arcus,tplot},scaling=constrained,
          tickmarks=[3,2],title="Polardarstellung komplexer Zahlen");
```

Polardarstellung komplexer Zahlen



r und φ lassen sich so berechnen:

```
> 'z' = z; phi := argument(z); r := abs(z);
```

$$z = 3 + 2\,i\,, \quad \varphi := \arctan\left(\frac{2}{3}\right), \quad r := \sqrt{13}$$

```
> 'phi' = evalf(%%,6);
```

$$\varphi = .588003$$

Wir überprüfen, daß dies wirklich die Polardarstellung von z liefert:

```
> 'r'*exp(I*'phi') = radnormal(evalc(r*exp(I*phi)));
```

$$r\,e^{i\,\varphi} = 3 + 2\,i$$

r und φ erhält man auch so:

```
> convert(z,polar);
```

$$\mathrm{polar}\left(\sqrt{13},\ \arctan\left(\frac{2}{3}\right)\right)$$

... oder so:

```
> polar(z);
```

$$\mathrm{polar}\left(\sqrt{13},\ \arctan\left(\frac{2}{3}\right)\right)$$

```
> r := op(1,%); phi := op(2,%%);
```

$$r := \sqrt{13}, \quad \varphi := \arctan\left(\frac{2}{3}\right)$$

Rechnen mit der Polardarstellung

```
> r := 'r': phi := 'phi': polar(r,phi): % = evalc(%);
  polar(r,phi)*polar(s,psi): % = simplify(%);
  1/polar(r,phi): % = simplify(%);
  polar(r,phi)^3: % = simplify(%);
```

$$\begin{aligned}
\mathrm{polar}(r,\,\varphi) &= r\cos(\varphi) + i\,r\sin(\varphi)\,,\\
\mathrm{polar}(r,\,\varphi)\,\mathrm{polar}(s,\,\psi) &= \mathrm{polar}(r\,s,\,\varphi+\psi)\,,\\
\frac{1}{\mathrm{polar}(r,\,\varphi)} &= \mathrm{polar}\left(\frac{1}{r},\,-\varphi\right),\\
\mathrm{polar}(r,\,\varphi)^3 &= \mathrm{polar}(r^3,\,3\,\varphi)
\end{aligned}$$

Wir machen uns zum Schluß klar, daß argument(z) der Hauptwert des Argumentes von z ist:

```
> 'argument(-1)' = argument(-1);
```

$$\mathrm{argument}(-1) = \pi$$

```
> Limit(argument(-1+I*t),t=0,left):  % = value(%);
  Limit(argument(-1+I*t),t=0,right): % = value(%);
```

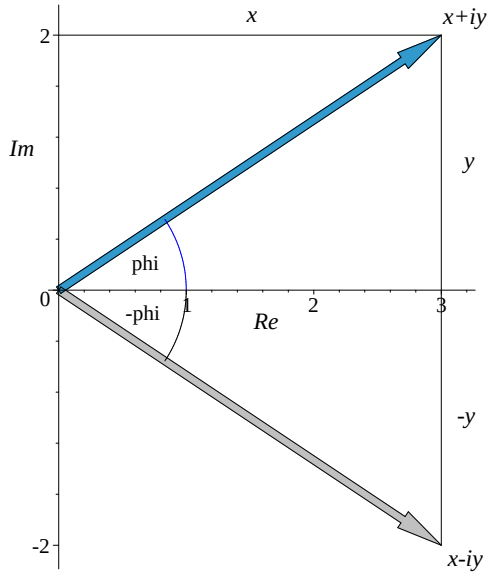
$$\lim_{t\to 0-} \mathrm{argument}(-1+i\,t) = -\pi\,, \quad \lim_{t\to 0+} \mathrm{argument}(-1+i\,t) = \pi$$

MWS 1

Veranschaulichung der Konjugation

```
> restart: with(plots): with(plottools):
  x := 3: y := 2: z := x+I*y: phi := argument(z):
  Pfeil1 := arrow([0,0],[x,y],0.06,0.15,0.1,color=cyan):
  Pfeil2 := arrow([0,0],[x,-y],0.06,0.15,0.1,color=gray):
  Arcus1 := arc([0,0],1,0..phi,color=blue,thickness=2):
  Arcus2 := arc([0,0],1,-phi..0,color=black,thickness=2):
  hLinie := line([0,y],[x,y],color=black):
  vLinie := line([x,-y],[x,y],color=black):
  Texte  := [x+0.2,y+0.15,"x+iy"],[x+0.2,-y-0.1,"x-iy"],
            [x/2,y+0.15," x"],[x+0.2,y/2," y"],[x+0.2,-y/2,"-y"],
            [0.7*cos(phi/2),0.6*sin(phi/2)+0.05," phi"],
            [0.7*cos(phi/2),-0.6*sin(phi/2),"-phi"]:
  tplot  := textplot([Texte]):
  display({Pfeil1,Pfeil2,Arcus1,Arcus2,hLinie,vLinie,tplot},
          tickmarks=[3,3],scaling=constrained,labels=[Re,Im],
          title="Konjugation komplexer Zahlen");
```

Konjugation komplexer Zahlen

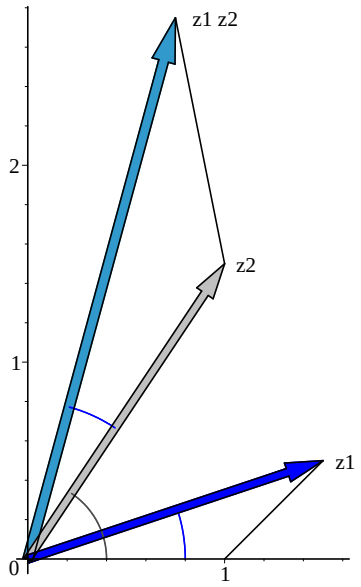


Veranschaulichung der Multiplikation

```
> restart: with(plots): with(plottools):
  x1 := 3/2: y1 := 1/2: z1 := x1+I*y1: phi1 := argument(z1):
  x2 := 1:   y2 := 3/2: z2 := x2+I*y2: phi2 := argument(z2):
  z := z1*z2: x := Re(z): y := Im(z):
  Pfeil1 := arrow([0,0],[x1,y1],0.04,0.1,0.12,color=blue):
```

```
Pfeil2 := arrow([0,0],[x2,y2],0.04,0.1,0.10,color=gray):
Pfeil3 := arrow([0,0],[x,y],0.05,0.12,0.08,color=cyan):
Arcus1 := arc([0,0],0.8,0..phi1,color=blue,thickness=2):
Arcus2 := arc([0,0],0.8,phi2..phi2+phi1,color=blue,thickness=2):
Arcus3 := arc([0,0],0.4,0..phi2,color=gray,thickness=2):
Linien := plot([[[1,0],[x1,y1]],[[x2,y2],[x,y]]],color=black):
Texte  := [x1+0.1,y1," z1"],[x2+0.1,y2," z2"],[x+0.2,y," z1 z2"]:
tplot  := textplot([Texte]):
display(Pfeil1,Pfeil2,Pfeil3,Arcus1,Arcus2,Arcus3,Linien,tplot,
        tickmarks=[2,3], scaling=constrained,
        title="Multiplikation komplexer Zahlen");
```

Multiplikation komplexer Zahlen



1.5 Wurzeln

Beispiel

```
> restart: interface(imaginaryunit=i): macro(I=i):
```

Es sollen die dritten Wurzeln von $w = 2 + 11\,i$ berechnet werden, d. h. wir suchen komplexe Zahlen z mit $z^3 = w$, die im Beispiel von Rafael Bombelli (siehe Seite 2) auftraten.

```
> w := 2+11*I;
```

$$w := 2 + 11\,i$$

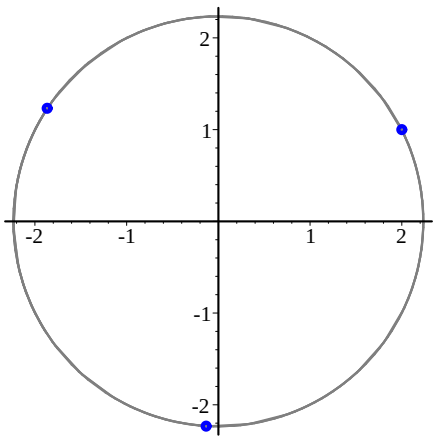
```
> solve(z^3=w); Loesungen := map(evalc,[%]);
```

$$(2+i)\left(-\frac{1}{2}+\frac{1}{2}i\sqrt{3}\right),\quad (2+i)\left(-\frac{1}{2}-\frac{1}{2}i\sqrt{3}\right),\quad 2+i$$

$$Loesungen := \left[-1-\frac{1}{2}\sqrt{3}+i\left(-\frac{1}{2}+\sqrt{3}\right),\ -1+\frac{1}{2}\sqrt{3}+i\left(-\frac{1}{2}-\sqrt{3}\right),\ 2+i\right]$$

Wir erhalten drei Lösungen, die in der folgenden Abbildung veranschaulicht sind:

```
> with(plots):
  dots   := complexplot(Loesungen,style=point,color=blue,
                        symbol=circle,symbolsize=16):
  Kreis := plot(abs(Loesungen[1]),coords=polar,color=gray,thickness=2):
  display(dots,Kreis,scaling=constrained);
```



Symbolische Berechnung n-ter Wurzeln

```
> restart: interface(imaginaryunit=i): macro(I=i):
  assume(r > 0):  a := r*exp(I*psi);
```

$$a := r\,e^{i\,\psi}$$

Für eine komplexe Zahl a mit der Polardarstellung $a = r\,e^{i\,\psi}$ berechnen wir die n-ten Wurzeln:

```
> solve(z^n=a,z): evalc(%): simplify(%);
```

$$r^{\frac{1}{n}}\left(\cos\left(\frac{\psi}{n}\right)+i\sin\left(\frac{\psi}{n}\right)\right)$$

Maple liefert symbolisch rechnend zunächst nur *eine* Lösung, obwohl es n Lösungen gibt. Wenn wir *alle Lösungen* haben wollen, müssen wir zuvor eine *Umgebungsvariable* geeignet setzen:

```
> _EnvAllSolutions := true:
  solve(z^n=a,z): evalc(%): simplify(%);
```

$$z := r^{\frac{1}{n}} \left(\cos\left(\frac{\psi + 2\,\pi\,\_Z1}{n} \right) + i \sin\left(\frac{\psi + 2\,\pi\,\_Z1}{n} \right) \right)$$

Wir wissen schon: Das sind nur scheinbar unendlich viele Lösungen. Unter diesen sind die für die Parameterwerte $\_Z1 = 0, 1, \ldots, n-1$ paarweise verschieden. Im Falle $a = 1$ bezeichnet man die n-ten Wurzeln als *Einheitswurzeln*. Sie sind die Lösungen der *Kreisteilungsgleichung* $z^n = 1$ und liefern die Eckpunkte eines regelmäßigen n-Ecks.

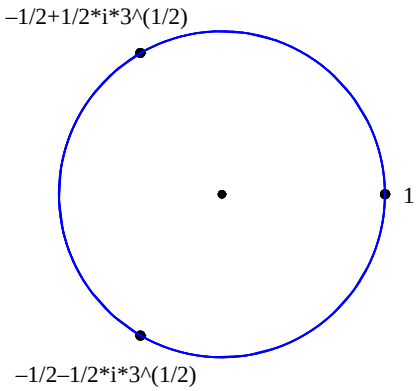
Beispiel

```
> n := readstat("Geben Sie eine natürliche Zahl ein: "):
```

Geben Sie eine natürliche Zahl ein: 3;

```
> W := [seq(2*Pi/n*k,k=0..n-1)]:    # Winkel
  P := [seq(exp(I*phi),phi=W)]:     # komplexe Punkte
  Q := map(z -> [Re(z),Im(z)],P):   # kartesische Darstellungen von P
  with(plots): with(plottools):
  Kreis := plot(1,coords=polar,color=blue,thickness=2):
  dots  := seq(disk(z,0.025,color=black),z=Q),
               disk([0,0],0.02,color=black):
  Text  := textplot([seq([op(evalf(1.2*z,3)),z[1]+I*z[2]],z=Q)]):
  display(Kreis,dots,Text,axes=none,scaling=constrained,
          title=cat(n,". Einheitswurzeln"));
```

3. Einheitswurzeln



Nach der Eingabe einer *beliebigen* natürlichen Zahl und deren Übergabe an die Variable n wird obige Abbildung erzeugt, welche die n-ten Einheitswurzeln veranschaulicht. Die hierzu ausgeführte Anweisungsgruppe wollen

wir ein wenig erläutern und zugleich daran deutlich machen, daß Maple-Code recht kompakt und dennoch übersichtlich sein kann. Die Liste P enthält alle n-ten Einheitswurzeln; deren Winkel (im Bogenmaß) sind in der Liste W aufgezählt. Wir weisen an dieser Stelle auf die Anweisung `seq(exp(I*phi), phi=W)` hin, die gewiß eleganter ist als die Formulierung seq(exp(I*W[k]), k=1..n) in der herkömmlichen Art. Die sehr wichtige Maple-Funktion `map` bildet mittels $z \to \big[\mathfrak{Re}(z), \mathfrak{Im}(z)\big]$ die komplexen Zahlen der Liste P auf die entsprechenden reellen Zahlenpaare der Liste Q ab. Natürlich hätten wir ebensogut

```
P := map(phi -> exp(I*phi),W): Q := [seq([Re(z),Im(z)],z=P)]:
```

schreiben können; welche Formulierung man bevorzugt, ist reine Geschmacksache.

Nach diesen Vorbereitungen werden die Graphikstrukturen *Kreis, dots* und *Text* definiert. *Kreis* ist offensichtlich der Einheitskreis, und *dots* stellt mit Hilfe des Kommandos *disk* aus dem Plottools-Paket die Einheitswurzeln sowie den Ursprung als ‚fette' Punkte dar. Ab Maple 6 kann man auch einfacher z. B. `dots:=plot(Q, style=point, symbol=circle, symbolsize=18);` schreiben. *Text* schließlich liefert mit Hilfe von *textplot* die Beschriftung durch eine Liste von Textangaben. Auf den ersten Blick wirkt der entsprechende Teil der Maple-Anweisung etwas kryptisch; wir empfehlen deshalb, folgende Anweisung separat auszuführen:

```
> seq([op(evalf(1.2*z,3)),z[1]+I*z[2]],z=Q);
```

$$[1.2,\ 0.,\ 1]\,,\ \left[-.600,\ 1.04,\ -\frac{1}{2}+\frac{1}{2}\,i\,\sqrt{3}\right],\ \left[-.600,\ -1.04,\ -\frac{1}{2}-\frac{1}{2}\,i\,\sqrt{3}\right]$$

Dann dürfte klar werden, was im einzelnen passiert. Der Faktor 1.2 soll übrigens bewirken, daß die Beschriftung vom Einheitskreis etwas abgesetzt wird. Und noch etwas sei an einem Beispiel erläutert:

```
> Q[2]; evalf(1.2*Q[2],3); op(%):
```

$$\left[\frac{-1}{2},\ \frac{1}{2}\,\sqrt{3}\right],\quad [-.600\,,\ 1.04]$$

Hier wird also das geordnete Paar $Q[2]$ mit dem Faktor 1.2 multipliziert und anschließend mittels `evalf(·,3)` auf drei Stellen gerundet, was im Rahmen der Zeichengenauigkeit ausreicht. Mit `op(%)` schließlich greift man auf die beiden Komponenten des geordneten Paares zu; dies sind genau die Koordinaten, welche in die zweite Textangabe oben eingesetzt worden sind. Die Ausgabe dieser drei Plotstrukturen erfolgt — wie so oft — mit *display*. Die Überschrift der Abbildung wird mittels der Option *title* erzeugt. Hier lernen wir erstmals die Maple-Funktion `cat` kennen, die n und „. Einheitswurzel" zu einem String verknüpft.

1.6 Riemannsche Zahlenkugel und stereographische Projektion

Bernhard RIEMANN verdanken wir ein Modell der komplexen Zahlen, welches sehr leicht eine Veranschaulichung des Punktes ∞ erlaubt. Die komplexen Zahlen werden dabei mit der (x_1, x_2)-Ebene des dreidimensionalen Raumes identifiziert. Punkte $z = x + i\,y$ der komplexen Zahlenebene werden auf die Einheitssphäre

$$S := \left\{(x_1, x_2, x_3) \in \mathbb{R}^3 \,|\, x_1^2 + x_2^2 + x_3^2 = 1\right\}$$

abgebildet, indem man den Schnittpunkt $Z = (x_1, x_2, x_3)$ von S mit der Geraden durch den Nordpol $N = (0, 0, 1)$ und $P = (x, y, 0)$ berechnet. Man erhält so die folgende Abbildung:

$$x_1 = \frac{2x}{x^2+y^2+1}\,, \quad x_2 = \frac{2y}{x^2+y^2+1}\,, \quad x_3 = \frac{x^2+y^2-1}{x^2+y^2+1}$$

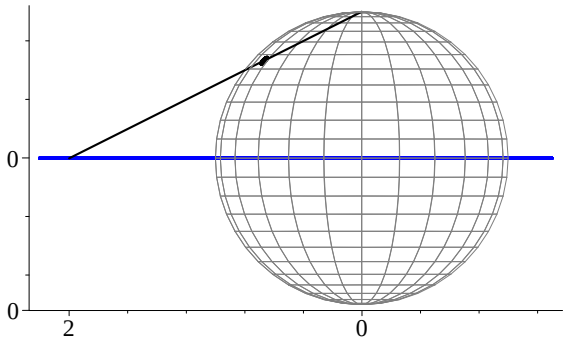
Für $z = 2+i$ wird die stereographische Projektion durch die Abbildung auf Seite 12 veranschaulicht. Diese wurde folgendermaßen mit Maple erzeugt:

```
> restart: with(plots): with(plottools):
  f := transform((x,y) -> [2*x,2*y,x^2+y^2-1]/(1+x^2+y^2)):
  g := transform((x,y) -> [x,y,0]):
  Punkt := disk([2,1],0.1,color=black):
  Ebene := plot3d(0,x=-1.3..2.2,y=-1.3..1.3,style=patchnogrid,
                  color=cyan):
  Gitter := plot3d(0,x=-1.3..2.2,y=-1.3..1.3,style=wireframe,
                   color=gray,numpoints=100):
  Kugel := plot3d(1,theta=0..2*Pi,phi=0..Pi,coords=spherical,
                  style=wireframe,color=gray,thickness=2):
  Gerade := line([2,1,0],[0,0,1],color=black,thickness=3):
  p := display(f(Punkt),g(Punkt),Kugel,Ebene,Gitter,Gerade,
               labels=[" "," " ," "]):
  display(p,axes=framed,title="stereographische Projektion",
          scaling=constrained,orientation=[-10,75],tickmarks=[2,2,2]);
```

Auf Einzelheiten der Graphikbausteine *Punkt, Ebene, Gitter, Kugel* und *Gerade* gehen wir nicht mehr näher ein; wir empfehlen jedoch, diese zum besseren Verständnis einzeln auszugeben und eventuell Optionen zur Oberflächengestaltung wie z. B. `style = patchnogrid` oder `style = wireframe` zu ändern oder gar wegzulassen. Besondere Beachtung verdient der Befehl `transform` aus dem Plottools-Paket. Die damit definierten Graphiktransformationen f und g wandeln hier 2D- in 3D-Graphikstrukturen. Näheres hierzu ist im Anhang zu finden.

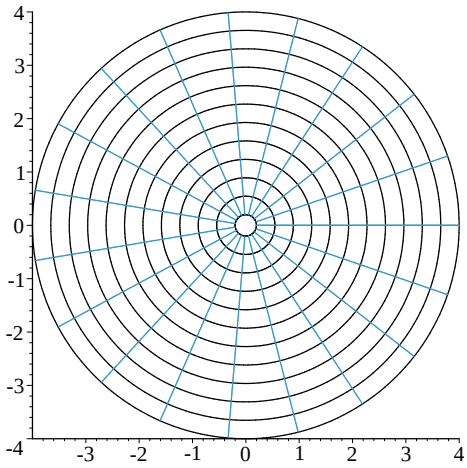
Mittels der Option `orientation = [90, 90]` erhalten wir folgende *Seitenansicht:*

```
> display(p,axes=framed,orientation=[90,90],tickmarks=[2,2,2],
          scaling=constrained);
```



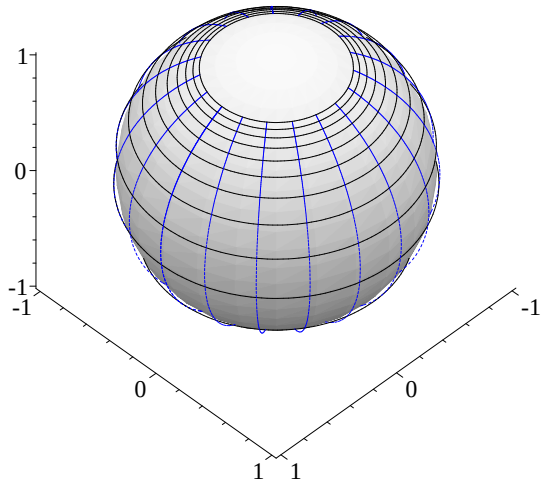
Jetzt soll das *Abbildungsverhalten der stereographischen Projektion* an komplexeren Graphikstrukturen verdeutlicht werden. Wir wählen ein *Polargitter*, das sehr einfach mit dem Kommando coordplot erzeugt werden kann. Wir erläutern die vier wichtigsten Parameter: polar ist der Name des gewünschten Koordinatensystems. Es folgt eine Liste von zwei Bereichen, die in der Parameterebene das Rechteck $[0.2, 4]\times[0, 2\,\pi]$ beschreiben; wegen grid $=$ [12, 20] muß man sich dieses durch ein rechteckiges Gitter von 12 (schwarzen) vertikalen bzw. 20 (blauen) horizontalen Linien überzogen denken. Durch view schließlich wird im kartesischen Koordinatensystem des Bildbereiches ein Fenster für das Bildgitter festgelegt.

```
> p := coordplot(polar,[0.2..4,0..2*Pi],view=[-4..4,-4..4],
                 grid=[12,20],thickness=2,color=[blue,black]):
  display(p,axes=framed,scaling=constrained);
```



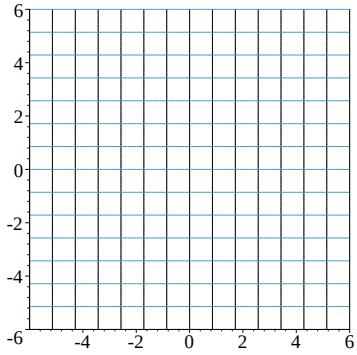
Zur besseren Visualisierung unterlegen wir die Plots mit einer Kugel um $(0, 0, 0)$, deren Radius etwas kleiner ist als der der Riemannschen Zahlenkugel:

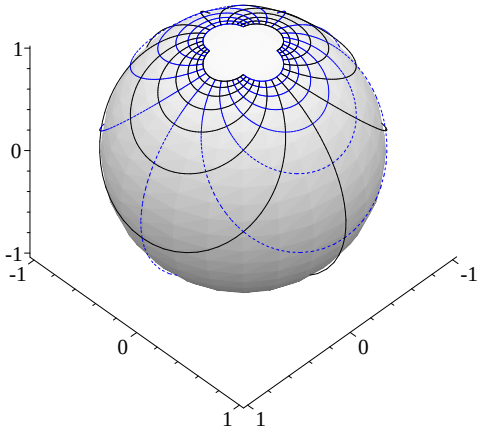
```
> Kugel1 := plot3d(0.98,theta=0..2*Pi,phi=0..Pi,shading=zgrayscale,
                   style=patchnogrid,coords=spherical,scaling=constrained):
  display(Kugel1,f(p),axes=framed,tickmarks=[3,3,3]);
```



Als zweites Beispiel wählen wir ein *Rechteckgitter* (kartesisches Gitter) und gehen analog vor:

```
> c := coordplot(cartesian,[-6..6,-6..6],view=[-6..6,-6..6],
                 grid=[15,15],thickness=2,color=[blue,black]):
  display(c,axes=framed,scaling=constrained);
  display(Kugel1,f(c),axes=framed,tickmarks=[3,3,3]);
```



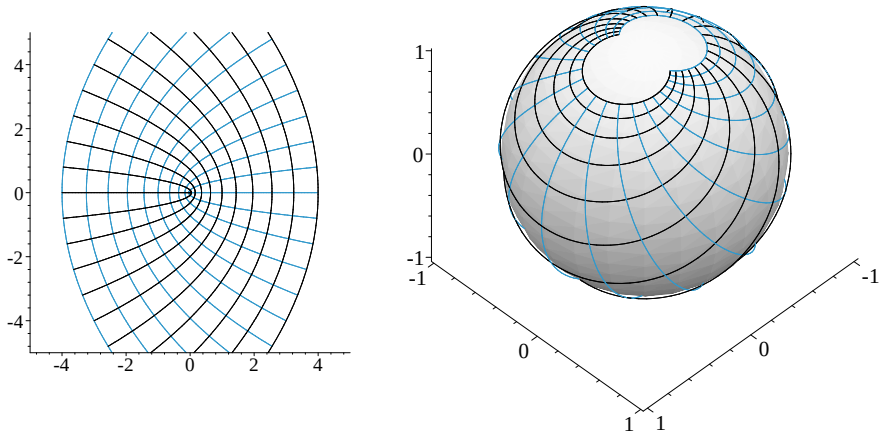


Und weil's so schön ist, noch zwei weitere Beispiele:

```
> sq := conformal(z^2,z=-2-2*I..2+2*I,grid=[21,21],numxy=[150,150],
                  thickness=2):
  display(sq,view=[-5..5,-5..5],axes=framed,scaling=constrained);
```

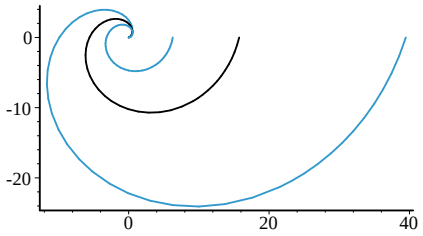
Wir erläutern die Bedeutung der drei wichtigsten Parameter:
Durch z = - 2 - 2∗I .. 2 + 2∗I wird in der komplexen Zahlenebene das (quadratische) Rechteck $[-2,\,2]\times[-2,\,2]$ beschrieben. Dieses wird wegen grid = [21, 21] — ähnlich wie bei coordplot — durch ein Raster von je 21 (blauen) horizontalen bzw. (schwarzen) vertikalen Gitterlinien überzogen. Durch die komplexe Abbildung $z\longmapsto z^2$ — aufgefaßt als reelle Abbildung $(x,\,y)\longmapsto[x^2-y^2,\,2\,x\,y]$ des $\mathbb{R}^2$ in sich — entsteht dann das Bildgitter. Es folgt die stereographische Projektion dieser 2D-Grafik:

```
> display(Kugel1,f(sq),axes=framed,tickmarks=[3,3,3]);
```



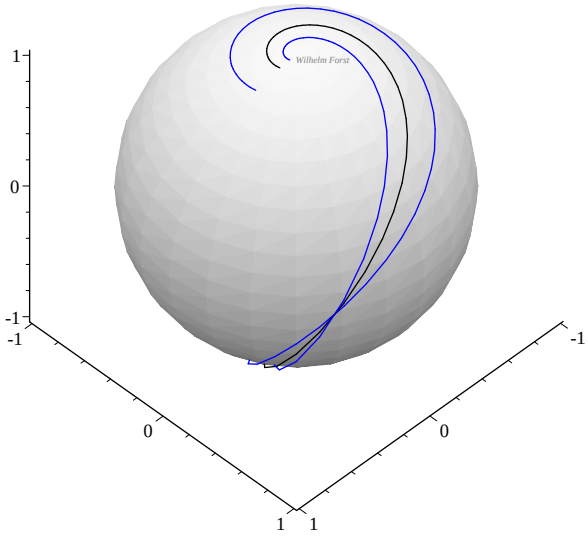
Wir beschließen den Reigen schöner Maple-Graphiken:

```
> s := plot([t,t^1.5,t^2],t=0..2*Pi,coords=polar,tickmarks=[3,3],
            color=[blue,black],thickness=2):
  display(s,axes=framed,scaling=constrained);
```




```
> display(Kugel1,f(s),axes=framed,title="Gusti, Lili und Modche ...",
          tickmarks=[3,3,3]);
```

Gusti, Lili und Modche ...



Das Bild dieser drei Kurven auf der Riemannschen Zahlenkugel assoziiert Erinnerungen an *Gusti, Lili und Modche* aus Ephraim Kishons Satire *Tagebuch eines Haarspalters*.

Leser, die sich über den nur auf der Kugel sichtbaren Kurvenschnittpunkt wundern, sollten in der Anweisung für die 2D-Graphikstruktur s den Bereich $t = 0 .. 2*\mathrm{Pi}$ etwa durch $t = 0 .. 1.1$ ersetzen.