Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 10/01

Die komplexen Wurzeln aus 1

Neundorf, Werner

Januar 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 01/10

Die komplexen Wurzeln aus 1

Werner Neundorf

Januar 2010

[‡]MSC (2000): 65-01, 65-05, 65H04, 65H10, 97F50

Zusammenfassung

Gegenstand dieser Arbeit sind die komplexen Wurzeln aus der Zahl Eins.
Dazu stellen wir einige ihrer Eigenschaften dar, betrachten die zugehörige Nullstellenaufgabe, untersuchen verschiedene numerische Verfahren zu ihrer Bestimmung und erkennen dabei interessante Zusammenhänge zu verwandten Gebieten.
Nach nützlichen Vorbetrachtungen erfolgen die Berechnungen sowie die zahlreichen graphischen Darstellungen der Funktionen, der Algorithmen und Situationen unter Verwendung der Computeralgebrasysteme Maple und MATLAB.

Carl Friedrich Gauß 1799
in seiner Doktorarbeit
über den Hauptsatz der Algebra

Quantumque scio nemo dubium contra hanc rem movit. Attamen si quis postulat, demonstrationem nullis dubiis obnoxiam alia occasione tradere suscipiam.

Soviel ich weiß, hat dies niemand je in Zweifel gezogen. Sollte es dennoch jemand fordern, so will ich zu anderer Gelegenheit einen unzweifelhaften Beweis erbringen.

Albrecht Beutelspacher

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

Inhaltsverzeichnis

Kapitel 1

Grundlagen

1.1 Der Hauptsatz der Algebra

Der Hauptsatz der Algebra, nach C.F. Gauss auch Fundamentalsatz der Algebra genannt, gibt Auskunft über die Lösbarkeit von Polynomgleichungen. Er besagt schlicht und ergreifend: Jedes komplexe Polynom vom Grad m hat m komplexe Nullstellen [12], [13]. Man sagt dazu auch, der Körper der komplexen Zahlen ist algebraisch abgeschlossen. Dieser Satz wird oft benutzt, zitiert und gelehrt. Sein Beweis und das Verständnis seiner Grundlagen sind ein Meilenstein der Mathematik des 19. Jahrhunderts. Er ist auch heute noch aktuell, zum Beispiel im Hinblick auf seine algorithmischen und numerischen Aspekte.
Seit Gauss' Dissertation 1799 wurden zahlreiche Beweise des Hauptsatzes der Algebra entwickelt, wahlweise mit Hilfsmitteln der Analysis, der Algebra oder der Topologie. Der Hauptsatz verbindet Analysis und Algebra auf bemerkenswerte Weise. Die Körper der reellen Zahlen $\mathbb{R}$ und der komplexen Zahlen $\mathbb{C}$ begegnen uns zuerst in der Analysis, für die sie die unentbehrliche Grundlage bilden. Die Aussage des Satzes hingegen ist algebraischer Natur, da sie nur von Polynomen handelt.

Satz 1.1 Hauptsatz der Algebra

Sei $\mathbb{C}$ der Körper der komplexen Zahlen.
Dann gilt: Für jedes Polynom m-ten Grades ($m \geq 1$)

$$p_m(z) = z^m + c_1 z^{m-1} + ... + c_{m-1} z + c_m \tag{1.1}$$

mit komplexen Koeffizienten $c_k,\ k = 1, 2, ..., m$, existieren komplexe Zahlen $z_k,\ k = 1, 2, ..., m$, so dass gilt

$$p_m(z) = (z - z_1)(z - z_2) \cdot ... \cdot (z - z_m). \tag{1.2}$$

Insbesondere interessiert uns hier das Polynom $p_m(z) = z^m - 1$.

Definition 1.1 Einheitswurzeln

Als Einheitswurzeln bezeichnen wir die komplexen Wurzeln z_k eines speziellen Polynom m-ten Grades mit reellen Koeffizienten und notieren diesen Sachverhalt als Nullstellenaufgabe für die Funktion

$$f(z) = z^m - 1 = (z - z_1)(z - z_2) \cdot \ldots \cdot (z - z_m),\ z = x + \imath\, y,\ \imath = \sqrt{-1}. \quad (1.3)$$

Da es sich bei den Einheitswurzeln um spezielle Werte handelt, werden wir dafür die Bezeichnung ω_k verwenden.
Man erkennt sofort, dass 1 eine Wurzel ist, für gerade Potenz m ebenfalls die Zahl -1 und wegen $f(z) = 0 = \overline{f(z)} = f(\bar{z})$ zu jeder Wurzel auch ihre konjugiert komplexe Größe.
Die Wurzeln von $f(z)$ sind auch genau die Wurzeln der Betragsfunktion $|f(z)|$. Hingegen wird die Menge der Lösungen der Gleichung $f(|z|) = |z|^m - 1 = 0$, was gleichbedeutend mit $|z| = 1$ ist, also der Einheitskreis sein und somit die m Einheitswurzeln enthalten.

Man betrachte die folgenden Maple Anweisungen zur Funktion $f(z)$ unter dem Aspekt ihrer Ergebisse, der Ausführung und eventuell dabei auftretenden Probleme.

```
> m:='m':
  f:=unapply(z^m-1,z);
  f(z);
  f(x+I*y);
  # analog
  # expand(%) assuming x::real, y::real;
  # expand(%%) assuming m::posint;
  # expand(%%%) assuming x::real, y::real, m::posint;
```

$$f := z \to z^m - 1$$
$$z^m - 1$$
$$(x + \mathrm{I}y)^m - 1$$

```
> solve(f(z));
  solve(f(x+I*y),{x,y});
```

$$\{m = m, z = 1\},\ \{m = 0, z = z\}$$
$$\{x = 1 - \mathrm{I}y, y = y\}$$

```
> # spezielle Faelle
  m:=3;
  f(z);
  f(x+I*y);
  expand(%);
```

$$m := 3$$
$$z^3 - 1$$
$$(x + \mathrm{I}y)^3 - 1$$
$$x^3 + 3\mathrm{I}x^2y - 3xy^2 - \mathrm{I}y^3 - 1$$

```
> solve(f(z));
  solve(f(x+I*y),{x,y}); # solve(f(x+I*y),{x,y}) assuming x::real,y::real;
```

$$1,\ -\frac{1}{2}+\frac{1}{2}\mathrm{I}\sqrt{3},\ -\frac{1}{2}-\frac{1}{2}\mathrm{I}\sqrt{3}$$

$$\{x=1-\mathrm{I}y, y=y\},\ \left\{x=-\frac{1}{2}-\mathrm{I}y+\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\},\ \left\{x=-\frac{1}{2}-\mathrm{I}y-\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\}$$

```
> m:=4;
  f(z);
  f(x+I*y);
  expand(%);
```

$$m := 4$$
$$z^4-1$$
$$(x+\mathrm{I}y)^4-1$$
$$x^4+4\mathrm{I}x^3y-6x^2y^2-4\mathrm{I}xy^3+y^4-1$$

```
> solve(f(z));
  solve(f(x+I*y),{x,y});
```

$$1,\ -1,\ \mathrm{I},\ -\mathrm{I}$$
$$\{x=-\mathrm{I}y-1, y=y\},\ \{x=1-\mathrm{I}y, y=y\},\ \{x=-\mathrm{I}y-\mathrm{I}, y=y\},\ \{x=-\mathrm{I}y+\mathrm{I}, y=y\}$$

```
> m:=5;
  f(z);
  f(x+I*y);
  expand(%);
```

$$m := 5$$
$$z^5-1$$
$$(x+\mathrm{I}y)^5-1$$
$$x^5+5\mathrm{I}x^4y-10x^3y^2-10\mathrm{I}x^2y^3+5xy^4+\mathrm{I}y^5-1$$

```
> solve(f(z));
  solve(f(x+I*y),{x,y});
```

$$1,\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}}$$
$$\{x=1-\mathrm{I}y, y=y\},\ \{x=$$
$$RootOf(\_Z^4+(1+4\mathrm{I}y)\_Z^3+(1-6y^2+3\mathrm{I}y)\_Z^2+(1-4\mathrm{I}y^3-3y^2+2\mathrm{I}y)\_Z+\mathrm{I}y+1+y^4-\mathrm{I}y^3-y^2), y=y\}$$

```
> m:=6;
  f(z);
  f(x+I*y);
  expand(%);
```

$$m := 6$$
$$z^6-1$$
$$(x+\mathrm{I}y)^6-1$$
$$x^6+6\mathrm{I}x^5y-15x^4y^2-20\mathrm{I}x^3y^3+15x^2y^4+6\mathrm{I}xy^5-y^6-1$$

```
> solve(f(z));
  solve(f(x+I*y),{x,y});
```

$$-1,\ 1,\ -\frac{1}{2}\sqrt{-2+2\mathrm{I}\sqrt{3}},\ \frac{1}{2}\sqrt{-2+2\mathrm{I}\sqrt{3}},\ -\frac{1}{2}\sqrt{-2-2\mathrm{I}\sqrt{3}},\ \frac{1}{2}\sqrt{-2-2\mathrm{I}\sqrt{3}}$$
$$\{x=-\mathrm{I}y-1, y=y\},\ \{x=1-\mathrm{I}y, y=y\},\ \left\{x=\frac{1}{2}-\mathrm{I}y+\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\},\ \left\{x=\frac{1}{2}-\mathrm{I}y-\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\},$$
$$\left\{x=-\frac{1}{2}-\mathrm{I}y+\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\},\ \left\{x=-\frac{1}{2}-\mathrm{I}y-\frac{1}{2}\mathrm{I}\sqrt{3}, y=y\right\}$$

```
> # Ableitung f'(z)
  m:='m':
  fs:=D(f);
  fs(z);
  simplify(%);
```

$$fs := z \to \frac{z^m m}{z}$$
$$\frac{z^m m}{z}$$
$$z^{m-1} m$$

```
> # Betrag |f(z)|=|z^m-1|
  bf:=unapply(abs(z^m-1),z);
  bf(z);
  # Ableitung
  bfs:=D(bf);
  bfs(z);          # diff(bf(z),z);
  simplify(%);
  # nicht auswertbar
  bfs(2);
  bfs(1);
```

$$bf := z \to |z^m - 1|$$
$$|z^m - 1|$$
$$bfs := z \to \frac{\mathrm{abs}(1, z^m - 1) z^m m}{z}$$
$$\frac{\mathrm{abs}(1, z^m - 1) z^m m}{z}$$
$$\mathrm{abs}(1, z^m - 1) z^{m-1} m$$
$$\frac{1}{2}\mathrm{abs}(1, 2^m - 1) 2^m m$$

```
Error, (in simpl/abs) abs is not differentiable at 0
```

```
> # |f(z)|=|z^m-1|=|(x+y*I)^m-1|
  af:=unapply(abs((x+y*I)^m-1),x,y);
  af(x,y);
```

$$af := (x, y) \to |(x + \mathrm{I}y)^m - 1|$$
$$|(x + \mathrm{I}y)^m - 1|$$

1.2 Einheitswurzeln und Eigenschaften

Wir werden im Weiteren die m-ten Einheitswurzeln einfach angeben und sofort feststellen, dass diese Größen die Gleichung $f(z) = 0$ mit $f(z)$ aus (1.3) erfüllen.
Dazu folgen noch einige Eigenschaften.

Definition 1.2 Referenz
Als Referenz bezeichnen wir die Folge von m äquidistanten Punkten (Stützstellen) im Intervall $[0, 2\pi]$

$$x_k \;=\; \frac{2\pi k}{m},\;\; k = 0, 1, ..., m - 1. \tag{1.4}$$

Definition 1.3 m-te Einheitswurzeln
Die m-ten Einheitswurzeln sind

$$\omega_k = e^{\imath\, x_k} = e^{\imath\, 2\pi k/m} = \cos(\tfrac{2\pi k}{m}) + \imath\, \sin(\tfrac{2\pi k}{m}),\;\; k = 0, 1, 2, ..., m - 1. \tag{1.5}$$

Satz 1.2 Eigenschaften der Einheitswurzeln

Für die m-ten Einheitswurzeln $\omega_k = e^{\imath\, x_k} = e^{\imath\, 2\pi k/m}$ *gelten die folgenden Eigenschaften*

$$\omega_k^0 = \omega_k^m = 1,\ \omega_k^j = \omega_j^k, \tag{1.6}$$

$$\omega_k^j = \omega_k^{j \bmod m}, \tag{1.7}$$

$$\omega_k^{-1} = e^{-\imath\, 2\pi k/m} = e^{\imath\, 2\pi(m-k)/m} = \overline{e^{\imath\, 2\pi k/m}}, \tag{1.8}$$

$$\sum_{k=0}^{m-1} \omega_k^j \omega_k^{-l} = m\, \delta_{jl}, \tag{1.9}$$

$$\sum_{k=0}^{m-1} \omega_k^j = \begin{cases} m & \textit{für } j = 0, \\ 0 & \textit{für } j = 1, 2, ..., m-1. \end{cases} \tag{1.10}$$

Beweis.

(1) Die Formeln (1.6) - (1.8) sind offensichtlich.

(2) Wegen $\omega_k^j \omega_k^{-l} = \omega_k^{j-l},\ \omega_k^j = \omega_j^k,\ \omega_k^0 = \omega_k^m = 1$ ist nur zu zeigen, dass

$$\sum_{k=0}^{m-1} \omega_j^k \ = \ m\, \delta_{0j}.$$

Es gilt $0 = \omega_j^m - 1 = (\omega_j - 1)(\omega_j^{m-1} + \omega_j^{m-2} + ... + 1)$.

Eine Fallunterscheidung ergibt

(a) $j \neq 0$: $\omega_j \neq 1 \ \Rightarrow \ \sum_{k=0}^{m-1} \omega_j^k = 0 = m\, \delta_{0j}$,

(b) $j = 0$: $\sum_{k=0}^{m-1} \omega_0^k = \sum_{k=0}^{m-1} 1 = m = m\, \delta_{00}$.

(3) Die Beziehung (1.10) folgt aus (1.6) sowie (1.9) mit $l = 0$. □

Maple Anweisungen zur Graphik

Zunächst fassen wir die Bilder von 6 Wurzelfällen in einem Tableau (Array) zusammen. In einer zweiten Version werden die Einzelbilder als seperate ps-Dateien abgespeichert und neben- und untereinander angeordnet.

```
> pp:=array(1..2,1..3,[ ]):
> mmax:=6:
> for m from 1 to mmax do
   sk:=[seq([cos(k/m*2*Pi),sin(k/m*2*Pi)],k=0..m-1)];
   pl1:=pointplot(sk,color=black,symbol=solidcircle,symbolsize=16):
   pl2:=pointplot([[1.2,0],[0,1.2],[-1.2,0],[0,-1.2]],color=white):
```

```
    pl3:=polygon(sk,color=yellow,filled=true):
    pl4:=implicitplot(x^2+y^2=1,x=-1..1,y=-1..1,color=blue,thickness=2):
    pl5:=plot([seq([[0,0],sk[k]],k=1..m)],color=black,style=line,
              linestyle=dot,thickness=2):
    pl6:=textplot([seq([1.1*op(sk[k]),`\omega`],k=1..m)],font=[TIMES,11]):
    pl7:=textplot([seq([1.1*(op(sk[k+1])+(0.06,-0.05)),`k`],k=0..m-1)],
              font=[TIMES,8]):
    display(pl1,pl2,pl3,pl4,pl5,pl6,pl7,
              view=[-1.2..1.2,-1.2..1.2],
              tickmarks=[3,3],axesfont=[TIMES,10],
              labels=[`Re(z)`,`Im(z)`],labelfont=[HELVETICA,9],
              title=cat(` m=`,`m`),titlefont=[TIMES,BOLD,12]);
    pp[1+iquo(m,4),1+irem(m-1,3)]:=display(pl1,pl2,pl3,pl4,pl5,pl6,pl7,
              view=[-1.2..1.2,-1.2..1.2],
              tickmarks=[3,3],axesfont=[TIMES,10],
              labels=[`Re(z)`,`Im(z)`],labelfont=[HELVETICA,9],
              title=cat(` m=`,`m`),titlefont=[TIMES,BOLD,12]):
  end do:

> display(pp);
> interface(plotdevice=ps,plotoutput=file00,
            plotoptions=`portrait,noborder,width=350,height=260`);
  plots[display](pp);
  interface(plotdevice=default);
```

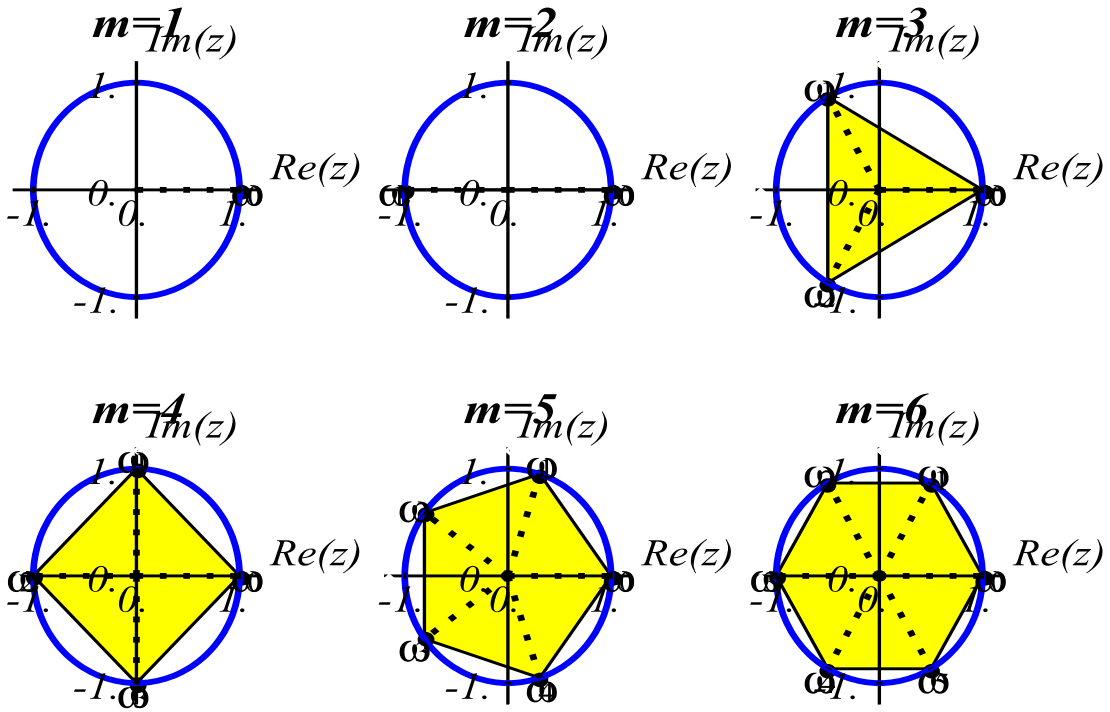


Abb. 1.1 Array-Plot,
Einheitswurzeln ω_k, $k = 0(1)m-1$, für $m = 1(1)6$ auf dem Einheitskreis

```
> display(pp[1,1]);
  interface(plotdevice=ps,plotoutput=file01,
            plotoptions='portrait,noborder,width=350,height=260');
  plots[display](pp[1,1]);
  interface(plotdevice=default);
  # analog die weiteren Bilder
```

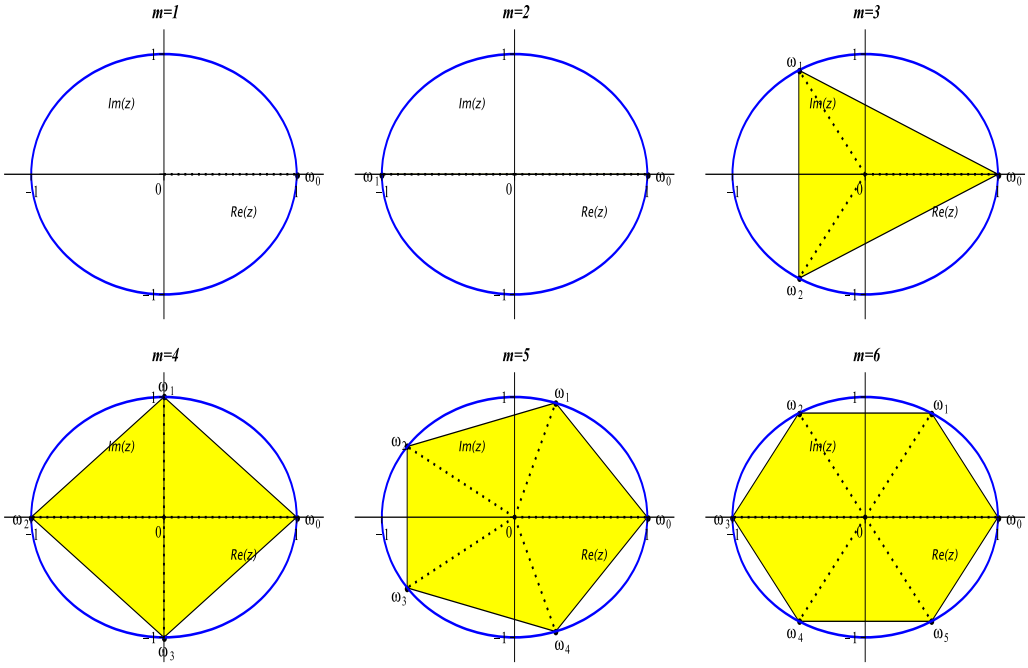


Abb. 1.2 Einzelbilder-Plot,
Einheitswurzeln $\omega_k,\ k = 0(1)m-1$, für $m = 1(1)6$ auf dem Einheitskreis

Man beachte, dass zwar die Abbildungen der Bildschirmanzeige des Array-Plots mit den Einzelbildern übereinstimmen, aber Maple beim Export des Array als ps-File die Plot-Optionen und einige Beschriftungen eigenständig anders wählt und einheitlich festlegt. Leider geschieht das in den meisten Fällen abweichend von den eingestellten Optionen und damit den Eigenschaften in der Einzelbilddarstellung. Die Qualität und Anschaulichkeit des Array-Plots leiden darunter.
Man sollte also für ein Array die Plot-Optionen, die Bezeichnungen und ähnliches modifizieren bzw. verändern. Hier sind die Plot-Optionen auf die Einzelbilder zugeschnitten.
Zum Beispiel wird in den obigen Maple Anweisungen der "künstlich" eingefügte Plotbefehl

```
> pl2:=pointplot([[1.2,0],[0,1.2],[-1.2,0],[0,-1.2]],color=white):
```

dafür verwendet, das im Array alle 6 Abbildungen den gleichen Achsenbereich haben. Die `view`-Option wird nämlich dort beim Graphikexport ignoriert.

Die Potenzen der Einheitswurzeln $\omega_k,\ k = 0, 1, ..., m-1$, kann man als Spalten in eine Matrix eintragen. Sei

$$A = \begin{pmatrix} 1 & \omega_0 & \omega_0^2 & \cdots & \omega_0^{m-2} & \omega_0^{m-1} \\ 1 & \omega_1 & \omega_1^2 & \cdots & \omega_1^{m-2} & \omega_1^{m-1} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots & & & & \\ 1 & \omega_{m-2} & \omega_{m-2}^2 & \cdots & \omega_{m-2}^{m-2} & \omega_{m-2}^{m-1} \\ 1 & \omega_{m-1} & \omega_{m-1}^2 & \cdots & \omega_{m-1}^{m-2} & \omega_{m-1}^{m-1} \end{pmatrix}. \tag{1.11}$$

Die Matrix A ist die reguläre Vandermondesche Matrix.
Als Inverse erhält man

$$A^{-1} = \frac{1}{m} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ \omega_0^{-1} & \omega_1^{-1} & \omega_2^{-1} & \cdots & \omega_{m-2}^{-1} & \omega_{m-1}^{-1} \\ \omega_0^{-2} & \omega_1^{-2} & \omega_2^{-2} & \cdots & \omega_{m-2}^{-2} & \omega_{m-1}^{-2} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & & & \\ \omega_0^{-(m-2)} & \omega_1^{-(m-2)} & \omega_2^{-(m-2)} & \cdots & \omega_{m-2}^{-(m-2)} & \omega_{m-1}^{-(m-2)} \\ \omega_0^{-(m-1)} & \omega_1^{-(m-1)} & \omega_2^{-(m-1)} & \cdots & \omega_{m-2}^{-(m-1)} & \omega_{m-1}^{-(m-1)} \end{pmatrix}, \tag{1.12}$$

$(\omega_0^k = \omega_k^0 = 1)$.

Die Matrizen spielen bei der trigonometrischen Interpolation und Fourier-Analyse eine wichtige Rolle. Dort braucht man auch die Funktionen

$$\Psi_j(x) = e^{\imath jx}. \tag{1.13}$$

Diese bilden ein Orthonormalsystem mit der Referenz $\{x_k\}$ und dem komplexen Skalarprodukt

$$(f, g) = \frac{1}{m} \sum_{k=0}^{m-1} f(x_k)\overline{g(x_k)}, \tag{1.14}$$

denn es gilt

$$(\Psi_j, \Psi_l) = (e^{\imath jx}, e^{\imath lx}) = \frac{1}{m} \sum_{k=0}^{m-1} e^{\imath jx_k}\, \overline{e^{\imath lx_k}} = \frac{1}{m} \sum_{k=0}^{m-1} \omega_k^j \omega_k^{-l} = \delta_{jl}. \tag{1.15}$$

1.3 Die Maple Kommandos `piecewise` und `if`

Im Zusammenhang mit Umrechnungen zwischen den Polarkoordinaten (r, φ) und den kartesischen Koordinaten (x, y), der Formel $z = x + \imath\, y = r\, e^{\imath\, \varphi} = |z|\, e^{\imath\, \mathrm{Arg}(z)}$, der Eulerschen Identität $e^{\imath\, \varphi} = \cos(\varphi) + \imath\, \sin(\varphi),\ r = 1$, sowie der Eulerschen Formel $e^z = e^{x+\imath\, y} = e^x\, e^{\imath\, y} = e^x\, (\cos(y) + \imath \sin(y))$ werden bedingte Anweisungen gebraucht. Dazu wollen wir die Funktionsweise der Kommandos `piecewise` und `if then else` testen.

(1) Einfacher Test

```
> f1:=x->piecewise(x<=1,1,x<=2,2,x<=3,3);  # default for otherwise is 0
  f1(0), f1(1.5), f1(2.5), f1(3.5);
  # analog zu f1
  f2:=x->piecewise(x<=1,1,x<=2,2,x<=3,3,0):
  f2(0), f2(1.5), f2(2.5), f2(3.5):
  f3:=x->piecewise(x<=1,1,1<x and x<=2,2,2<x and x<=3,3,3<x,0):
  f3(0), f3(1.5), f3(2.5), f3(3.5):
  # anders, nicht sinnvoll
  f4:=x->piecewise(x>1,2,x>2,3,x>3,0,1);
  f4(0), f4(1.5), f4(2.5), f4(3.5);
```

$$x \rightarrow piecewise(x \leq 1,\ 1,\ x \leq 2,\ 2,\ x \leq 3,\ 3)$$
$$1,\ 2,\ 3,\ 0$$
$$x \rightarrow piecewise(x > 1,\ 2,\ x > 2,\ 3,\ x > 3,\ 0,\ 1)$$
$$1,\ 2,\ 2,\ 2$$

(2) Skalarer (eindimensionaler) Fall

```
> x=0;
  x<0;
  0<x and x<2;
  x<=0 or x>=2;
  x=0 and x>-2;   # Anteil x=0 wird als false interpretiert
                  # und macht Ergebnis false
  x=0 or x>=2;    # Anteil x=0 wird als false interpretiert
                  # und "gestrichen"
  x=0 or x=2;     # beide Anteile als false interpretiert
  # Ausweg
  Or(x=0,x>=2);   # nicht or(x=0,x>=2)
  Or(x=0,x=2);
```

$$x = 0$$
$$x < 0$$
$$0 < x \text{ and } x < 2$$
$$x \leq 0 \text{ or } 2 \leq x$$
$$false$$
$$2 \leq x$$
$$false$$
$$\mathrm{Or}(x = 0,\ 2 \leq x)$$
$$\mathrm{Or}(x = 0,\ x = 2)$$

Die Verknüpfung einer Gleichheitsbedingung mit einer anderen führt zunächst dazu, dass diese Bedingung als `false` verwertet wird. Das Ergebnis des logischen Ausdrucks hängt dann vom Wahrheitswert der zweiten Bedingung ab.
Einen möglichen Ausweg bietet hier das Kommando `Or`.

```
> # Problem bei x=0
  zf:=unapply(piecewise(x=0 or x>=2,-1,0<x and x<2,1/x,exp(-x^2-1)),x);
  zf(0);          # x=0 wie otherwise interpretiert
  zf(-1), zf(1), zf(2), zf(3);
```

$$x \to piecewise\Big(2 \le x,\ -1,\ 0 < x \text{ and } x < 2,\ \frac{1}{x},\ e^{-x^2-1}\Big)$$

$$e^{-1}$$

$$e^{-2},\ 1,\ -1,\ -1$$

```
> # Korrekte Variante 1
  zr1:=unapply(piecewise(x=0,-1,x>=2,-1,0<x and x<2,1/x,exp(-x^2-1)),x);
  zr1(0);
  zr1(-1), zr1(1), zr1(2), zr1(3);
> # Korrekte Variante 2
  zr2:=unapply(piecewise(Or(x=0,x>=2),-1,0<x and x<2,1/x,exp(-x^2-1)),x):
  zr2(0):
  zr2(-1), zr2(1), zr2(2), zr2(3):
```

$$x \to piecewise\Big(x = 0,\ -1,\ 2 \le x,\ -1,\ 0 < x \text{ and } x < 2,\ \frac{1}{x},\ e^{-x^2-1}\Big)$$

$$-1$$

$$e^{-2},\ 1,\ -1,\ -1$$

(3) Mehrdimensionaler Fall
Die Situation ändert sich nicht im Vergleich zu Punkt (2). Es treten einige Fälle mehr auf. Man beachte die Kommentare und vergleiche die Ergebnisse.

```
> x>=0 and y>0;
  x>0 or y<=4;
  x=0 and 1>0;   # Anteil x=0 wird als false interpretiert
                 # und macht Ergebnis false
  x=0 and y>0;
  x=0 and y=0;
  x=0 or 1>0;    # Anteil x=0 wird als false interpretiert
                 # und "gestrichen"
  x=0 or y>0;
  x=0 or y=0;
  # Ausweg
  And(x=0,1>0);  # wie x=0,       nicht and(x=0,1>0)
  And(x=0,y>0);
  And(x=0,y=0);
  Or(x=0,1>0);   # wie 1>0=true,  nicht or(x=0,1>0)
  Or(x=0,y>0);
  Or(x=0,y=0);
```

$$0 \le x \text{ and } 0 < y$$

$$0 < x \text{ or } y \le 4$$

$$false$$

$$false$$

$$false$$

$$true$$

$$0 < y$$

$$false$$

$$\mathrm{And}(x = 0,\ 0 < 1)$$

$$\mathrm{And}(x = 0,\ 0 < y)$$

$$\mathrm{And}(x = 0,\ y = 0)$$

$$\mathrm{Or}(x = 0,\ 0 < 1)$$

$$\mathrm{Or}(x = 0,\ 0 < y)$$

$$\mathrm{Or}(x = 0,\ y = 0)$$

```
> # Problem bei x=0,y=0
  zf:=unapply(piecewise(x=0 and y=0,1,x<>0,2*y^2/x,-1),x,y);
  zf(0,0);       # x=0,y=0 wie otherwise interpretiert
  zf(1,0), zf(0,1), zf(1,1);
```

$$(x,y) \to piecewise\Big(x \neq 0,\ \frac{2y^2}{x},\ -1\Big)$$

$$-1$$

$$0,\ -1,\ 2$$

```
> # Korrekte Variante
  zr:=unapply(piecewise(And(x=0,y=0),1,x<>0,2*y^2/x,-1),x,y);
  zr(0,0);
  zr(1,0), zr(0,1), zr(1,1);
```

$$(x,y) \to piecewise\Big(\mathrm{And}(x=0,\ y=0),\ 1,\ x \neq 0,\ \frac{2y^2}{x},\ -1\Big)$$

$$1$$

$$0,\ -1,\ 2$$

(4) Funktionsdefinition, Auswertung und Graphik
Hier betrachten wir die Fälle `piecewise` und `if then else`

```
> f1:=x->piecewise(x<0,1,x<1,x^2+1,2*ln(x)+2);
  # f2:=unapply(piecewise(x<0,1,x<1,x^2+1,2*ln(x)+2),x);  # analog
  f1(x);
  f1(1);
  plot(f1(x),x=-1..3,thickness=2,view=[-1..3,0..4]):      # o.k.
```

$$x \to piecewise(x < 0,\ 1,\ x < 1,\ x^2+1,\ 2\ln(x)+2)$$

$$piecewise(x < 0,\ 1,\ x < 1,\ x^2+1,\ 2\ln(x)+2)$$

$$2$$

```
> f3:=x->if x<0 then 1 elif x<1 then x^2+1 else 2*ln(x)+2 end if;
  f3(x);
  f3(1);
  plot(f3(x),x=-1..3,thickness=2,view=[-1..3,0..4]);
```

$$x \to \text{if } x < 0 \text{ then } 1 \text{ elif } x < 1 \text{ then } x^2+1 \text{ else } 2\ln(x)+2 \text{ end if}$$

```
Error, (in f3) cannot determine if this expression is true or false: x < 0
```

$$2$$

```
Error, (in f3) cannot determine if this expression is true or false: x < 0
```

Analoge Rück- bzw. Fehlermeldungen wie bei `f3` liefern auch die folgenden Prozeduren zusammen mit den Kommandos

```
fi(x); fi(1); plot(fi(x),x=-1..3,thickness=2,view=[-1..3,0..4]);
```

```
> f4:=proc(x)
      if x<0 then 1 elif x<1 then x^2+1 else 2*ln(x)+2 end if;
  end proc;
```

```
> f5:=proc(x)
      local y;
      if x<0 then y:=1 elif x<1 then y:=x^2+1 else y:=2*ln(x)+2 end if;
      y;
  end proc;
```

1.4 Kartesische und Polarkoordinaten

Die Umrechnungsformeln zwischen den ebenen Polarkoordinaten (Kreiskoordinaten) (r, φ) und den kartesischen Koordinaten (x, y) sind allgemein bekannt und folgendermaßen.

$$\begin{aligned} x &= r\cos(\varphi), \\ y &= r\sin(\varphi), \end{aligned} \tag{1.16}$$

$$r = \sqrt{x^2 + y^2},$$

$$\varphi = \begin{cases} 0 & \text{für } x = 0,\ y = 0, \\ \frac{\pi}{2} & \text{für } x = 0,\ y > 0, \\ \frac{3\pi}{2} & \text{für } x = 0,\ y < 0, \\ \arctan(\frac{y}{x}) & \text{für } x > 0,\ y \geq 0, \\ \arctan(\frac{y}{x}) + 2\pi & \text{für } x > 0,\ y < 0, \\ \arctan(\frac{y}{x}) + \pi & \text{für } x < 0. \end{cases} \tag{1.17}$$

Dem Koordinatenursprung (0,0) wird per Definition und zwecks Eindeutigkeit der Winkel $\varphi = 0$ zugeordnet.
Hier erfolgt die Berechnung des Winkels φ im Intervall $[0, 2\pi)$. Genauso kann man die Rechnung für das Intervall $(-\pi, \pi]$ machen, wie es bei der MATLAB Funktion `[phi,r]=cart2pol(x,y)` passiert.
Die Formel mit dem Arcuscosinus kommt unter Verwendung von r mit nur drei Fallunterscheidungen aus.

$$r = \sqrt{x^2 + y^2},$$

$$\varphi = \begin{cases} 0 & \text{für } r = 0, \\ \arccos(\frac{x}{r}) & \text{für } r > 0,\ y \geq 0, \\ 2\pi - \arccos(\frac{x}{r}) & \text{für } r > 0,\ y < 0. \end{cases}$$

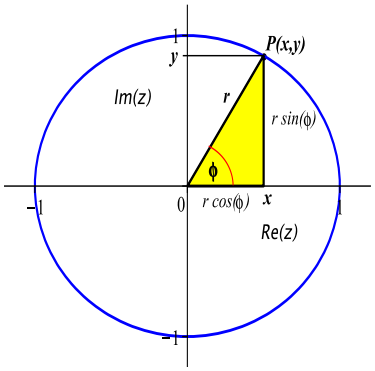


Abb. 1.3
Beziehungen
zwischen den kartesischen Koordinaten (x, y)
und den Polarkoordinaten (r, φ)

1.4.1 Maple-Anweisungen

(1) Prozeduren

```
> polar2kart1 := proc(r,phi)       # ev. r::nonnegative
   [r*cos(phi),r*sin(phi)];
  end proc:
> z:=polar2kart1(r,phi);
```

$$z := [r\,cos(\varphi),\ r\,sin(\varphi)]$$

```
> kart2polar1 := proc(x,y)
   local r,phi;
   r:=sqrt(x^2+y^2);
   if x=0 then if y=0 then phi:=0
                       elif y>0 then phi:=Pi/2 else phi:=3*Pi/2 end if
          else
            phi:=arctan(y/x);
            if x<0 then phi:=phi+Pi
                   elif y<0 then phi:=phi+2*Pi end if
   end if;
   [r,phi];
  end proc:
> z:=kart2polar1(x,y);
Error, (in kart2polar1) cannot determine if this expression is true or false: x < 0

> # guenstiger mit piecewise, And, Or
  kart2polar2 := proc(x,y)
   [sqrt(x^2+y^2),
    piecewise(And(x=0,y=0),0,And(x=0,y>0),Pi/2,And(x=0,y<0),3*Pi/2,
              x<0,arctan(y/x)+Pi,
              And(x>0,y<0),arctan(y/x)+2*Pi,
              arctan(y/x))];
  end proc:
> z:=kart2polar2(x,y);
```

$$z := \left[\sqrt{x^2+y^2},\ \begin{cases} 0 & \mathrm{And}(x=0,y=0) \\ \frac{1}{2}\pi & \mathrm{And}(x=0,0<y) \\ \frac{3}{2}\pi & \mathrm{And}(x=0,y<0) \\ \arctan\left(\frac{y}{x}\right)+\pi & x<0 \\ \arctan\left(\frac{y}{x}\right)+2\pi & \mathrm{And}(0<x,y<0) \\ \arctan\left(\frac{y}{x}\right) & otherwise \end{cases}\right]$$

(2) Polarkoordinaten → kartesische Koordinaten

```
> readstat();            # Halt und Eingabefenster
                         # dann mit OK bzw. Enter weiter
> while true do
   r:=readstat("r = ");
   if r>=0 then break; else fprintf(default,‘Achtung r>=0\n‘): end if
  end do:
  fprintf(default,‘Radius r =%6.3f\n‘,r):
```

```
Radius r = 1.000
```

```
> phi:=readstat("Winkel phi (in Gradmass)  = ");
  p:=phi*Pi/180:
  fprintf(default,‘Winkel phi (in Bogenmass) =%6.3f\n‘,p):
```

$$\varphi := 60$$

```
Winkel phi (in Bogenmass) = 1.047
```

```
> fprintf(default,‘1. Direkte Berechnung\n‘):
  x:=r*cos(p):
  y:=r*sin(p):
  fprintf(default,‘x = %10.6f,  y = %10.6f\n‘,x,y):
```

```
1. Direkte Berechnung
x =   0.500000,  y =   0.866025
```

```
> fprintf(default,‘2. Kein spezielles Kommando fuer (Winkel,Radius)
                      ->(x,y)\n‘):
  fprintf(default,‘   siehe Punkt 1\n‘):
```

```
2. Kein spezielles Kommando fuer (Winkel,Radius)->(x,y)
   siehe Punkt 1
```

```
> fprintf(default,‘3. Funktion  polar2kart1:=proc(r,phi)\n‘):
  xy:=polar2kart1(r,p);
  x:=xy[1];              # polar2kart1(r,p)[1];
  y:=xy[2];              # polar2kart1(r,p)[2];
  fprintf(default,‘x = %10.6f,  y = %10.6f\n‘,x,y):
```

```
3. Funktion  polar2kart1:=proc(r,phi)
```

$$xy := \left[\frac{1}{2},\ \frac{1}{2}\sqrt{3}\right]$$
$$x := \frac{1}{2}$$
$$y := \frac{1}{2}\sqrt{3}$$

```
x =   0.500000,  y =   0.866025
```

(3) Kartesische Koordinaten → Polarkoordinaten

```
> x:=readstat("x = ");
  y:=readstat("y = ");
```

$$x := 1$$
$$y := 1$$

```
> fprintf(default,‘1. Direkte Berechnung fuer x,y>0\n‘):
  if x>0 and y>0 then
   r1:=sqrt(x^2+y^2):
   p1:=arctan(y/x):
   phi1:=180*p1/Pi:
   fprintf(default,‘r = %10.6f,  phi = %10.6f = %6.2f\n‘,r1,p1,phi1):
  end if:
```

```
1. Direkte Berechnung fuer x,y>0
r =   1.414214,  phi =   0.785398 =  45.00
```

```
> fprintf(default,‘2. Kommandos  polar(z), convert(z,polar), argument(z),
                      abs(z), z=x+yI\n‘):
  fprintf(default,‘   polar : convert to polar form, -Pi<Winkel<=Pi\n‘):
  z:=x+y*I;
  polar(z);              # convert(z,polar);
  p2:=argument(z);
  phi2:=180*p2/Pi;
  r2:=abs(z);
  fprintf(default,‘r = %10.6f,  phi = %10.6f = %6.2f\n‘,r2,p2,phi2):
```

```
2. Kommandos  polar(z), convert(z,polar), argument(z), abs(z), z=x+yI
   polar : convert to polar form, -Pi<Winkel<=Pi
```

$$x := 1+\mathrm{I}$$
$$\mathrm{polar}\left(\sqrt{2}, \frac{1}{4}\pi\right)$$
$$p2 := \frac{1}{4}\pi$$
$$\varphi 2 := 45$$
$$r2 := \sqrt{2}$$

```
r =   1.414214,  phi =   0.785398 =  45.00
```

```
> fprintf(default,‘3. Funktion  kart2polar1:=proc(x,y)\n‘):
  rp:=kart2polar1(x,y);
  r3:=rp[1];             # kart2polar1(x,y)[1];
  p3:=rp[2];             # kart2polar1(x,y)[2];
  phi3:=180*p3/Pi;
  fprintf(default,‘r = %10.6f,  phi = %10.6f = %6.2f\n‘,r3,p3,phi3):
```

```
3. Funktion  kart2polar1:=proc(x,y)
```

$$rp := \left[\sqrt{2}, \frac{1}{4}\pi\right]$$
$$r3 := \sqrt{2}$$
$$p3 := \frac{1}{4}\pi$$
$$\varphi 3 := 45$$

```
r =   1.414214,  phi =   0.785398 =  45.00
```

```
> fprintf(default,‘4. Funktion  kart2polar2:=proc(x,y)\n‘):
  rp:=kart2polar2(x,y);
  r4:=rp[1];             # kart2polar2(x,y)[1];
  p4:=rp[2];             # kart2polar2(x,y)[2];
  phi4:=180*p4/Pi;
  fprintf(default,‘r = %10.6f,  phi = %10.6f = %6.2f\n‘,r4,p4,phi4):
```

```
4. Funktion  kart2polar2:=proc(x,y)
```

$$rp := \left[\sqrt{2}, \frac{1}{4}\pi\right]$$
$$r4 := \sqrt{2}$$
$$p4 := \frac{1}{4}\pi$$
$$\varphi 4 := 45$$

```
r =   1.414214,  phi =   0.785398 =  45.00
```

1.4.2 MATLAB-Anweisungen

Es werden die Kommandos, Umrechnungsfunktionen in mehreren Versionen sowie das Skript-File mit seinen Ergebnissen aufgelistet.

(1) Kommandos

```
[x,y]=pol2cart(phi,r)
[phi,r]=cart2pol(x,y)        # -pi<phi<=pi
```

(2) Funktionen

```
% polar2kart1.m
% Polarkoordinaten  -> kartesische Koordinaten
% r,phi (Bogenmass) -> x,y
function [x,y]=polar2kart1(r,phi)
 x=r*cos(phi);
 y=r*sin(phi);
% end
```

```
% kart2polar1.m
% kartesische Koordinaten -> Polarkoordinaten
% x,y -> r,phi (0<=phi<2*pi, Bogenmass)
function [r,phi]=kart2polar1(x,y)
 r=sqrt(x^2+y^2);
 if x==0, if y==0, phi=0; elseif y>0, phi=pi/2; else phi=3*pi/2; end
   else
     phi=atan(y/x);
     if x<0, phi=phi+pi;
     elseif y<0, phi=phi+2*pi; end;
 end
%end

% kart2polar1a.m   Modifikation
% kartesische Koordinaten -> Polarkoordinaten
% x,y -> r,phi (0<=phi<2*pi, Bogenmass)
function [r,phi]=kart2polar1a(x,y)
 r=sqrt(x^2+y^2);
 if x==0, if y==0, phi=0; elseif y>0, phi=pi/2; else phi=3*pi/2; end
   elseif y==0, if x>0, phi=0; else phi=pi; end
     else
       phi=atan(y/x);
       if x<0, phi=phi+pi; elseif y<0, phi=phi+2*pi; end;
 end
%end
```

```
% kart2polar2.m
% kartesische Koordinaten -> Polarkoordinaten
% x,y -> r,phi (0<=phi<2*pi, Bogenmass)
function [r,phi]=kart2polar2(x,y)
 r=sqrt(x^2+y^2);
 p=1;
 switch p
  case x==0, switch p
              case y==0, phi=0;
              case y>0,  phi=pi/2;
              case y<0,  phi=3*pi/2;
             end
  otherwise, phi=atan(y/x);
             switch p
              case x<0, phi=phi+pi;
              case x>0 && y<0, phi=phi+2*pi;
             end
 end
%end

% kart2polar2a.m   Modifikation
% kartesische Koordinaten -> Polarkoordinaten
% x,y -> r,phi (0<=phi<2*pi, Bogenmass)
function [r,phi]=kart2polar2a(x,y)
 r=sqrt(x^2+y^2);
 p=1;
 switch p
  case x==0, switch p
              case y==0, phi=0;
              case y>0,  phi=pi/2;
              case y<0,  phi=3*pi/2;
             end
  otherwise, switch p
              case y==0, if x>0, phi=0; else phi=pi; end
              otherwise
               phi=atan(y/x);
               if x<0, phi=phi+pi; elseif y<0, phi=phi+2*pi; end;
             end
 end
%end
```

(3) Skript-File

```
% koordinaten1.m
% Umrechnung zwischen kartesischen und Polarkoordinaten
% (x,y) <-> (r,phi) (0<=phi<2*pi, Bogenmass, bzw. 0<=phi<360, Gradmass)
```

```
diary koordinaten1.txt    % Protokolldatei
diary on
echo off
clear all
clc  % clf
format compact
format short

fprintf('\nKartesische Koordinaten <-> Polarkoordinaten')
fprintf('\n_____________________________________________\n')
fprintf('\nPolarkoordinaten -> Kartesische Koordinaten \n')
while 1
  r=input('Radius r  = ');
  if r>=0, break; else disp('Achtung r>=0'); end
end
phi=input('Winkel phi (in Gradmass)  = ');
p=phi*pi/180;
fprintf('Winkel phi (in Bogenmass) =%6.3f\n',p);

fprintf('\n1. Direkte Berechnung\n');
x=r*cos(p);
y=r*sin(p);
fprintf('x = %10.6f,  y = %10.6f\n\n',x,y);
fprintf('2. Kommando [x,y]=pol2cart(phi,r)\n');
[x1,y1]=pol2cart(p,r);
fprintf('x = %10.6f,  y = %10.6f\n\n',x1,y1);
fprintf('3. Funktion [x,y]=polar2kart1(r,phi)\n');
[x2,y2]=polar2kart1(r,p);
fprintf('x = %10.6f,  y = %10.6f\n',x2,y2);
pause

fprintf('\nKartesische Koordinaten -> Polarkoordinaten \n')
x=input('x = ');
y=input('y = ');
fprintf('\n1. Direkte Berechnung fuer x,y>0\n')
if x>0 & y>0
  r=sqrt(x^2+y^2);
  p=atan(y/x);
  phi=180*p/pi;
  fprintf('r = %10.6f,  phi = %10.6f = %6.2f\n\n',r,p,phi);
end
fprintf('2. Kommando [phi,r]=cart2pol(x,y), -pi<phi<=pi\n');
[p1,r1]=cart2pol(x,y);
phi1=180*p1/pi;
fprintf('r = %10.6f,  phi = %10.6f = %6.2f\n\n',r1,p1,phi1);
fprintf('3. Funktion [phi,r]=kart2polar1(x,y)\n');
```

```
[r2,p2]=kart2polar1(x,y);
phi2=180*p2/pi;
fprintf('r = %10.6f,  phi = %10.6f = %6.2f\n\n',r2,p2,phi2);
fprintf('4. Funktion [phi,r]=kart2polar2(x,y)\n');
[r3,p3]=kart2polar2(x,y);
phi3=180*p3/pi;
fprintf('r = %10.6f,  phi = %10.6f = %6.2f\n\n',r3,p3,phi3);
pause
format
diary off
echo off
```

(4) Protokolldatei zu `koordinaten1.m`

```
Kartesische Koordinaten <-> Polarkoordinaten
--------------------------------------------

Polarkoordinaten -> Kartesische Koordinaten
Radius r  = 1
Winkel phi (in Gradmass)  = 60
Winkel phi (in Bogenmass) = 1.047

1. Direkte Berechnung
x =   0.500000,  y =   0.866025

2. Kommando [x,y]=pol2cart(phi,r)
x =   0.500000,  y =   0.866025

3. Funktion [x,y]=polar2kart1(r,phi)
x =   0.500000,  y =   0.866025

Kartesische Koordinaten -> Polarkoordinaten
x = 1
y = 1

1. Direkte Berechnung fuer x,y>0
r =   1.414214,  phi =   0.785398 =  45.00

2. Kommando [phi,r]=cart2pol(x,y), -pi<phi<=pi
r =   1.414214,  phi =   0.785398 =  45.00

3. Funktion [phi,r]=kart2polar1(x,y)
r =   1.414214,  phi =   0.785398 =  45.00

4. Funktion [phi,r]=kart2polar2(x,y)
r =   1.414214,  phi =   0.785398 =  45.00
```

Kapitel 2

Das Newton-Verfahren

2.1 Das Newton-Verfahren im $\mathbb{R}$

Das Newton-Verfahren, auch Newton-Raphson-Verfahren oder Tangentennäherungsverfahren genannt, zur näherungsweisen iterativen Bestimmung einer Lösung x^* (Nullstelle) von $f(x) = 0$ hat die Gestalt

$$x_{k+1} \;=\; x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, ..., \; x_0 \text{ gegeben.} \tag{2.1}$$

Es ist lokal quadratisch konvergent, besitzt also die Konvergenzordnung ist $p = 2$. Es existiert damit eine Konstante $Q > 0$ (Konvergenzfaktor), so dass für alle Indizes $k \geq k_0, \; k_0 \in \mathbb{N}$, für den Lösungsfehler $e_k = |x_k - x^*|$ die Ungleichung

$$e_{k+1} \;\leq\; Q\, e_k^p. \tag{2.2}$$

erfüllt. Man sagt auch, die Folge $\{e_k\}$ besitzt die Q-Ordnung p.
Praktisch heißt dies grob, dass sich bei $p = 2$ pro Iterationsschritt die Anzahl der genauen Mantissenstellen der Iterierten im Vergleich zur Lösung verdoppelt.
Der asymptotische Konvergenzfaktor ist

$$0 < q \;=\; \lim_{k\to\infty} \frac{e_{k+1}}{e_k^p} \;=\; \frac{1}{2}\left|\frac{f''(x^*)}{f'(x^*)}\right|. \tag{2.3}$$

Mit dem Konvergenzverhalten des Newton-Verfahrens betrachtet man auch den Bereich der Konvergenz zur Nullstelle. Man spricht auch von ihrem Einzugsbereich, also wo sie die Folge der Iterierten anzieht. Seine Größe hängt von den Eigenschaften der Funktion ab. Im Eindimensionalen sind das Intervalle, im $\mathbb{R}^2$ Gebiete, die sich auch graphisch geeignet darstellen lassen. In den Maple-Arbeitsblättern ist die Rechengenauigkeit `Digits:=20`, in den Prozeduren `Digits:=32` eingestellt.
Weitere Informationen entnehme man aus [1].

Maple-Prozedur zum klassischen Newton-Verfahren im Reellen mit fester Iterationsanzahl, Übergabe der letzten Näherungslösung sowie zusätzlich Ausgabe von Zwischenergebnissen und Rückgabe des gesamten Iterationsvektors (globale Größe `xv`).

```
> Newton_reell := proc(f::procedure,xstart::numeric,
                       maxiter::posint,aus::name)::numeric;
   local k,xold,xnew,df,fxold,fxnew,delta,Digits_old;
   global xv;
  Digits_old := Digits; Digits := 32;
  df := D(f);       # Ableitung von f
  xold := xstart;  fxold := evalf(f(xold));
  if aus=ja then
   fprintf(default,‘\nIterationsverlauf\n‘):
   fprintf(default,‘k = %2.0f   x = %+.16e,  f(x) = %+.3e\n‘,0,xold,fxold):
  end if;
  for k from 1 to maxiter do
   xnew := evalf(xold-fxold/df(xold));
   delta := abs(xold-xnew);
   fxnew := evalf(f(xnew));
   xv := xv,xnew;
   xold := xnew;
   fxold := fxnew;
   if aus=ja then
    fprintf(default,‘k = %2.0f   x = %+.16e,  f(x) = %+.3e\n‘,k,xold,fxold):
   end if;
  end do;
  if aus=ja then
   fprintf(default,‘delta =  %.20e\n\n‘,delta):
  end if;
  Digits := Digits_old;
  xnew;
  end proc:
```

Beispiel 2.1 Newton-Verfahren für $f(x) = x - \sin(x) - \frac{1}{4}$

k	x_k	$f(x_k)$
0	3	2.608e 00
1	1.689 000 085 981 757	4.459e-01
2	1.290 067 646 705 414	7.921e-02
3	1.180 496 469 799 805	5.701e-03
4	1.171 293 663 705 201	3.911e-05
5	1.171 229 655 590 455	1.887e-09
6	1.171 229 652 501 666	4.394e-18
7	1.171 229 652 501 666	-1.000e-32

Tab. 2.1 Iterationsfolge x_k mit Funktionswerten

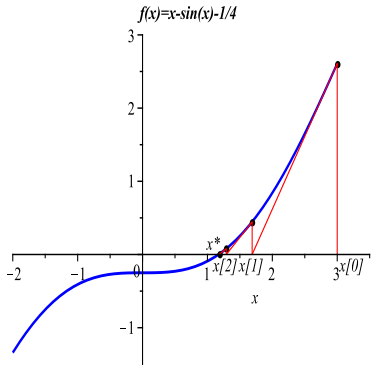


Abb. 2.1 Iterationsverlauf

Wegen $f'(0) = 0$ kann man $x_0 = 0$ nicht als Startwert nehmen. Tritt dieser Wert in der Iterationsfolge auf, bricht die Iteration ab. Bei numerischen Rechnungen ist das jedoch eher unwahrscheinlich, so dass das Einzugsintervall $\mathbb{R}\backslash\{0\}$ ist.

Analog sind die Formeln für das klassische Newton-Verfahren im Komplexen.

Maple-Prozedur zum Newton-Verfahren im Komplexen mit fester Iterationsanzahl, Übergabe der letzten Näherungslösung sowie zusätzlich Ausgabe von Zwischenergebnissen und Rückgabe des gesamten Iterationsvektors (globale Größe `zv`)

```
> Newton_komplex := proc(f::procedure,zstart::complex,
                         maxiter::posint,aus::name)::complex;
   local k,zold,znew,df,fzold,fznew,delta,Digits_old;
   global zv;

  Digits_old := Digits;
  Digits := 32;
  df := D(f);       # Ableitung von f
  zold := zstart;
  fzold := evalf(f(zold));
  if aus=ja then
   fprintf(default,‘\nIterationsverlauf\n‘);
   fprintf(default,
   ‘k = %2.0f   z = %+.16e %+.16ei,  f(z) = %+.3e %+.3ei, |f(z)|= %+.3e\n‘,
    0,Re(zold),Im(zold),Re(fzold),Im(fzold),abs(fzold));
  end if;
  for k from 1 to maxiter do
   znew := evalf(zold-fzold/df(zold));
   delta := abs(zold-znew);
   fznew := evalf(f(znew));
   zv := zv,znew;
   zold := znew;
   fzold := fznew;
   if aus=ja then
    fprintf(default,
    ‘k = %2.0f   z = %+.16e %+.16ei,  f(z) = %+.3e %+.3ei, |f(z)|= %+.3e\n‘,
     k,Re(zold),Im(zold),Re(fzold),Im(fzold),abs(fzold));
   end if;
  end do;
  if aus=ja then
   fprintf(default,‘delta =  %.20e\n\n‘,delta);
  end if;
  Digits := Digits_old;
  znew;
  end proc:
```

Beispiel 2.2 Newton-Verfahren für $f(z) = e^z - z$

Die Funktion hat im Reellen keine Nullstelle.
Neben der üblichen Vorgehensweise im Komplexen kann man $f(z) = 0,\ z = x + \imath y$, in ein nichtlineares Gleichungssystem der Dimension 2 überführen und dann geeignet weiterverarbeiten.
Das nichtlineare System ist

$$\begin{aligned} f_1(x,y) &= e^x \cos(y) - x &= 0, \\ f_2(x,y) &= e^x \sin(y) - y &= 0. \end{aligned}$$

Natürlich kann man noch nach x bzw. y umstellen und für jede der Variablen eine seperate Gleichung finden, z. B.

$$\begin{aligned} g(x) &= x - \frac{\arccos(xe^{-x})}{\tan(\arccos(xe^{-x}))} &= 0, \\ h(y) &= \frac{y}{\tan(y)} - \ln\left(\frac{y}{\sin(y)}\right) &= 0. \end{aligned}$$

Dabei sind möglichst äquivalente Umformungen zu machen und es ist darauf zu achten, ob und wie eventuelle Verluste an Lösungen/Informationen auftreten, z. B. hat $\arccos(x)$ nur den Wertebereich $[0, \pi]$.
Die Gleichung hat unendlich viele Lösungen, die positiven Realteil haben und noch symmetrisch zur x-Achse liegen.
Die ersten zehn Nullstellen sind

```
x = 0.318131505204764, y = +-1.337235701430689,
x = 2.062277729598284, y = +-7.588631178472513,
x = 2.653191974038697, y = +-13.94920833453321,
x = 3.020239708164501, y = +-20.27245764161522,
x = 3.287768611544093, y = +-26.58047149935914.
```

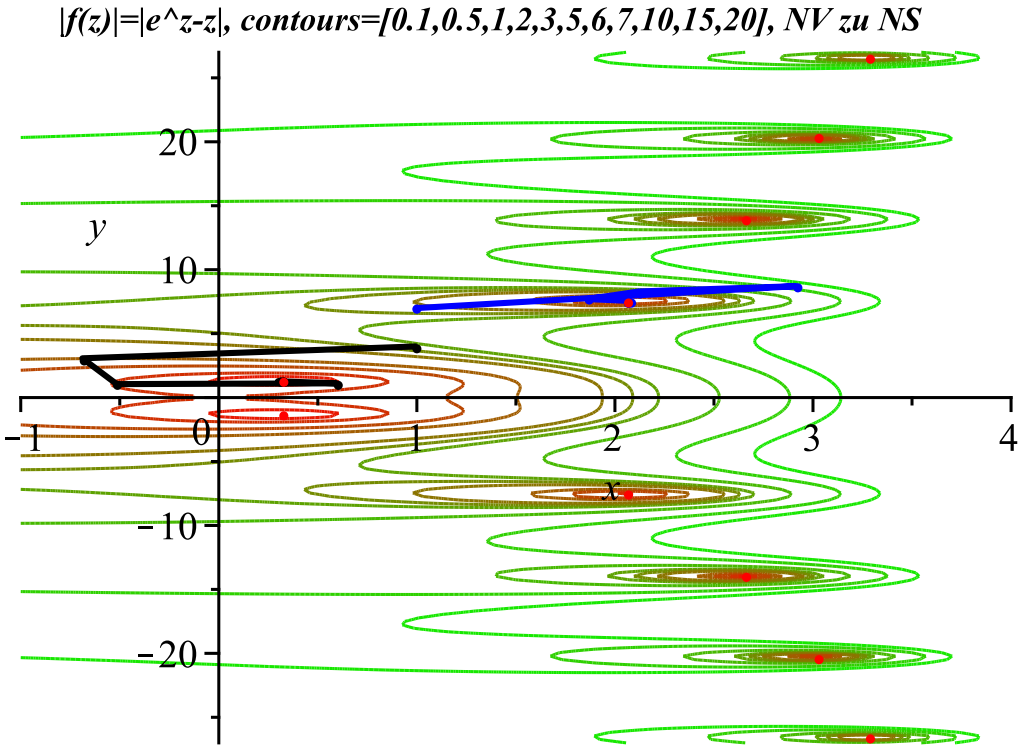


Abb. 2.2 Höhenlinien von $|f(z)|$ und Iterationsverläufe für zwei NS

Einige Aufrufe der Prozedur `Newton_komplex`
Abbruch bei Iteration im Reellen

```
> zstart:=1:
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,3,ja);
  ziter1:=[zv];
```

```
Iterationsverlauf
k =  0   z = +1.0000000000000000e+00 +0.0000000000000000e+00i,  f(z) = +1.718e+00 +0.000e+00i, |f(z)|= +1.718e+00
k =  1   z = +0.0000000000000000e+00 +0.0000000000000000e+00i,  f(z) = +1.000e+00 +0.000e+00i, |f(z)|= +1.000e+00
k =  2   z = -Inf +0.0000000000000000e+00i,  f(z) = +Inf +0.000e+00i, |f(z)|= +Inf
k =  3   z = +NaN +0.0000000000000000e+00i,  f(z) = +NaN +0.000e+00i, |f(z)|= +NaN
delta =  NaN
```

$$zs := Float(undefined)$$
$$ziter1 := [1, 0., -Float(\infty), Float(undefined)]$$

Iterationsverläufe zu Abbildung 2.2

```
> zstart:=1+4*I:
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,10,ja);
  ziter1:=[zv];
```

```
Iterationsverlauf
k =  0   z = +1.0000000000000000e+00 +4.0000000000000000e+00i,  f(z) = -2.777e+00 -6.057e+00i, |f(z)|= +6.663e+00
k =  1   z = -6.8902821403413046e-01 +3.0699578616712892e+00i,  f(z) = +1.883e-01 -3.034e+00i, |f(z)|= +3.040e+00
k =  2   z = -5.1528539575563771e-01 +1.0524816879125024e+00i,  f(z) = +8.112e-01 -5.336e-01i, |f(z)|= +9.710e-01
k =  3   z = +5.9332244081993529e-01 +1.1115950259843824e+00i,  f(z) = +2.089e-01 +5.109e-01i, |f(z)|= +5.520e-01
k =  4   z = +2.9851356364543033e-01 +1.2762949474715917e+00i,  f(z) = +9.272e-02 +1.353e-02i, |f(z)|= +9.370e-02
k =  5   z = +3.1768180396897081e-01 +1.3391322890520661e+00i,  f(z) = -2.229e-03 -1.897e-03i, |f(z)|= +2.927e-03
k =  6   z = +3.1812981730150428e-01 +1.3372361141412824e+00i,  f(z) = +5.990e-07 -2.539e-06i, |f(z)|= +2.608e-06
k =  7   z = +3.1813150520528471e-01 +1.3372357014294087e+00i,  f(z) = +1.358e-12 +1.569e-12i, |f(z)|= +2.075e-12
k =  8   z = +3.1813150520476414e-01 +1.3372357014306894e+00i,  f(z) = +6.737e-25 -1.128e-24i, |f(z)|= +1.314e-24
k =  9   z = +3.1813150520476414e-01 +1.3372357014306894e+00i,  f(z) = +2.000e-32 +0.000e+00i, |f(z)|= +2.000e-32
k = 10   z = +3.1813150520476414e-01 +1.3372357014306894e+00i,  f(z) = +1.000e-32 +0.000e+00i, |f(z)|= +1.000e-32
delta =  8.750661719816921936 73e-25
```

$$zs := 0.318131505204764135312654251587 66 + 1.337235701430689408901162143193 7I$$
$$ziter1 := [1 + 4I, -0.689028214034130458913660548035 0 + 3.069957861671289247895994330784 6I,$$
$$-0.515285395755637709279587212311 49 + 1.052481687912502350243506791723 7I,$$
$$0.593322440819935286426899059830 41 + 1.111595025984382378043155532623 1I,$$
$$...]$$

```
> zstart:=1+7*I:
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,10,ja);
  ziter2:=[zv];
```

```
Iterationsverlauf
k =  0   z = +1.0000000000000000e+00 +7.0000000000000000e+00i,  f(z) = +1.049e+00 -5.214e+00i, |f(z)|= +5.319e+00
k =  1   z = +2.9137304726222459e+00 +8.7120081841237902e+00i,  f(z) = -1.685e+01 +3.337e+00i, |f(z)|= +1.718e+01
k =  2   z = +2.1210730425557880e+00 +8.2960893823397672e+00i,  f(z) = -5.689e+00 -7.579e-01i, |f(z)|= +5.740e+00
k =  3   z = +1.8600822270594837e+00 +7.6995192797978179e+00i,  f(z) = -8.717e-01 -1.352e+00i, |f(z)|= +1.608e+00
k =  4   z = +2.0727765610402822e+00 +7.5618033001943233e+00i,  f(z) = +2.162e-01 +4.826e-02i, |f(z)|= +2.215e-01
k =  5   z = +2.0619338478781405e+00 +7.5883801459428815e+00i,  f(z) = +1.539e-03 -2.876e-03i, |f(z)|= +3.262e-03
k =  6   z = +2.0622777688759514e+00 +7.5886312628049152e+00i,  f(z) = -5.982e-07 +3.876e-07i, |f(z)|= +7.129e-07
k =  7   z = +2.0622777295982815e+00 +7.5886311784725164e+00i,  f(z) = -3.088e-14 -1.430e-14i, |f(z)|= +3.403e-14
k =  8   z = +2.0622777295982839e+00 +7.5886311784725126e+00i,  f(z) = +5.950e-29 -4.970e-29i, |f(z)|= +7.753e-29
delta =  4.44099188710719234783e-15
```

$$zs := 2.062277729598283884978486720002 7 + 7.588631178472512622568923954099 0I$$
$$ziter2 := [1 + 7I, ...]$$

2.2 Das Newton-Verfahren im $\mathbb{R}^n$

Betrachten wir das klassische Newton-Verfahren für ein System n nichtlinearer Gleichungen in seiner Normalform oder Nullstellenform

$$f(x) = 0, \quad f : \mathbb{R}^n \to \mathbb{R}^n, \quad f(x) = (f_1(x), f_2(x), ..., f_n(x))^T, \tag{2.4}$$

wobei der Lösungsvektor x^* Nullstelle von $f(x)$ ist.
Dazu brauchen wir die Jacobi-Matrix von f

$$\mathcal{J}(f(x)) = f'(x) = (f_{ij}(x)) = \left(\frac{\partial f_i(x)}{\partial x_j}\right)_{i,j=1}^{n}, \quad \text{Kurzform } \mathcal{J}(x). \tag{2.5}$$

Das Newton-Verfahren notieren wir in zwei Versionen, entweder als

$$x^{(k+1)} = x^{(k)} - [\mathcal{J}(x^{(k)})]^{-1} f(x^{(k)}), \quad k = 0, 1, 2, ..., \tag{2.6}$$

oder als in jedem Schritt $k = 0, 1, 2, ...$ zu lösendes lineares Gleichungssystem mit Korrektur der Iterierten

$$\mathcal{J}(x^{(k)})\, \delta^{(k)} = f(x^{(k)}), \quad x^{(k+1)} = x^{(k)} - \delta^{(k)}. \tag{2.7}$$

Die Korrektur $\delta^{(k)}$ heißt auch Newton-Richtung.

Beispiel 2.3 Man bestimme in der Umgebung des Vektors $(1.2, 1.7)^T$ die reelle Lösung $x^* = (x_1^*, x_2^*)^T = (1.234\,274\,484\,114\,475\,994,\ 1.661\,526\,466\,795\,933\,889)^T$ des nichtlinearen Gleichungssystems

$$\begin{aligned} f_1(x_1, x_2) &= 2x_1^3 - x_2^2 - 1 &&= 0, \\ f_2(x_1, x_2) &= x_1 x_2^3 - x_2 - 4 &&= 0. \end{aligned}$$

Der Startvektor sei $x^{(0)} = (1.2, 1.7)^T$.
Die Jacobi-Matrix ist

$$\mathcal{J}(x) = \begin{pmatrix} 6x_1^2 & -2x_2 \\ x_2^3 & 3x_1 x_2^2 - 1 \end{pmatrix}, \quad \det(\mathcal{J}(x^*)) \neq 0, \ \det(\mathcal{J}(0,0)) = 0.$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$f_1(x^{(k)})$	$f_2(x^{(k)})$
0	1.2	1.7	–4.340e-01	+1.956e-01
1	1.234 876 263 287	1.660 979 680 824	+7.320e-03	–2.283e-03
2	1.234 274 675 323	1.661 526 275 856	+2.382e-06	–8.838e-07
3	1.234 274 484 114	1.661 526 466 795	+2.343e-13	–7.807e-14
4	1.234 274 484 114	1.661 526 466 795	+2.379e-27	–8.563e-28

Tab. 2.2 Iterationsverlauf des Newton-Verfahrens mit $x^{(0)} = (1.2, 1.7)^T$ sowie $x^{(k)}$ und $f(x^{(k)})$, $k = 0, 1, ..., 4$

Betrachten wir uns das Höhenlinienbild der Koordinatenfunktionen $f_1(x)$ und $f_2(x)$. Es lässt vermuten, dass x^* in der Tat die einzige reelle Lösung ist.

```
> f:=[2*x1^3-x2^2-1, x1*x2^3-x2-4];

> pl1:=contourplot(f[1],x1=-4..4,x2=-4..4,grid=[161,161],color=red,
              thickness=3,contours=[0]):
  pl1a:=contourplot(f[1],x1=-4..4,x2=-4..4,grid=[161,161],
              coloring=[gold,red],thickness=1,
              contours=[-80,-40,-20,-8,-4,-2,-1.2,-1,-0.95,-0.5,
                        0.6782,2,5,15,40,80]):
  pl2:=contourplot(f[2],x1=-4..4,x2=-4..4,grid=[161,161],color=blue,
              thickness=3,contours=[0]):
  pl2a:=contourplot(f[2],x1=-4..4,x2=-4..4,grid=[161,161],
              coloring=[cyan,blue],thickness=1,
              contours=[-40,-20,-10,-5,-4.385,-4.2,-4,-3.8,-3.5,
                        -2,5,15,40,80]):
  pl3:=pointplot(xs,color=black,symbol=solidcircle,symbolsize=16):
  pl4:=textplot([1.5,1.9,`x*`],font=[HELVETICA,BOLD,10]):

> display(pl1,pl1a,pl2,pl2a,pl3,pl4,view=[-4..4,-4..4],
              title=`f(x)=0, Hoehenlinien zu f1(x),f2(x), NS x*`,
              titlefont=[TIMES,BOLD,9]);
```

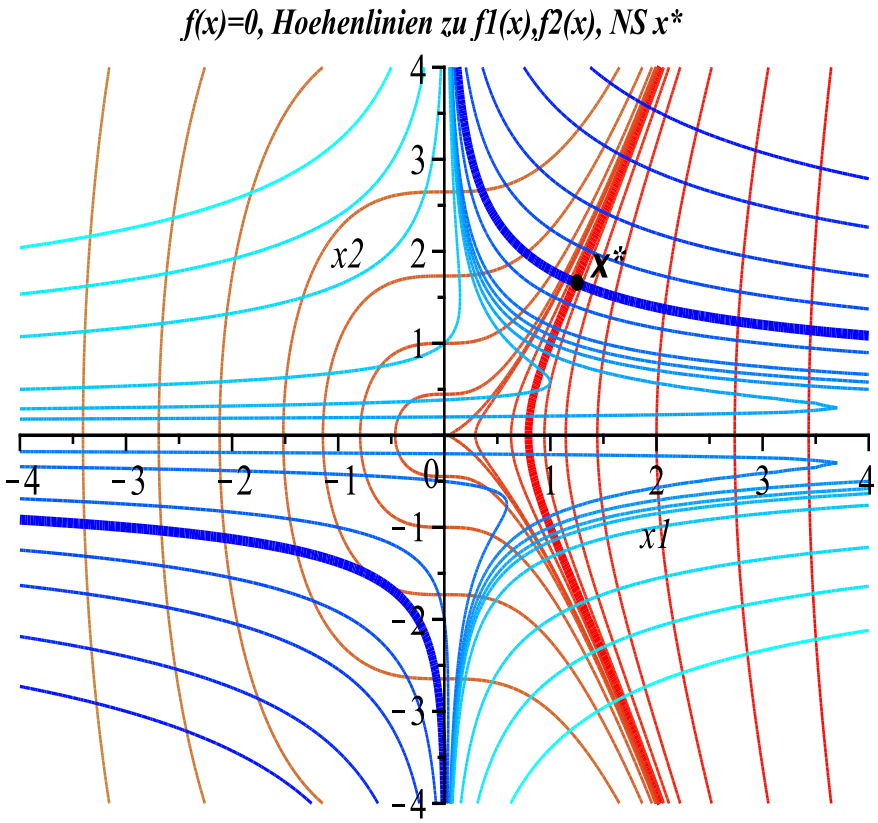


Abb. 2.3 Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit Lösung x^*, Nullniveaus dick hervorgehoben

Die Jacobi-Matrix ist im Koordinatenursprung singulär. Es gibt jedoch auch noch andere Vektoren, wo ihre Determinante

$$\det(\mathcal{J}(x)) = 6x_1^2(3x_1x_2^2 - 1) + 2x_2^4$$

verschwindet, z. B. ist $\left(1, \sqrt{(\sqrt{93} - 9)/2}\right)^T$ ein solcher Punkt. An diesen Stellen wird das Newton-Verfahren theoretisch abbrechen.
Aus der Menge $M = \{x : \det(\mathcal{J}(x)) = 0\}$ sollten weder Startvektoren verwendet werden, noch sollte die Iterationsfolge auf einen solchen Punkt treffen. Aber mit Rechnungen in der Gleitkommaarithmetik wird man an solchen Stellen "vorbeischlittern", außer beim Punkt $(0, 0)^T$ als Startvektor. Man merkt es bei der Verfolgung des Iterationsverlaufes, denn die Funktionswerte $f_{1,2}(x^{(k)})$ haben dann große Beträge. Aber irgendwann, auch wenn sehr viele Schritte zu machen sind, werden sich die Iterierten der Lösung x^* annähern. So kann man die numerischen Rechnungen eigentlich mit einem beliebigen Startvektor aus $\mathbb{R}^2\backslash\{(0, 0)^T\}$ beginnen. Günstige Startbedingungen findet man im oberen Teil des 1. Quadranten.
Der Einzugsbereich der Nullstelle x^* ist somit $\mathbb{R}^2\backslash M$.

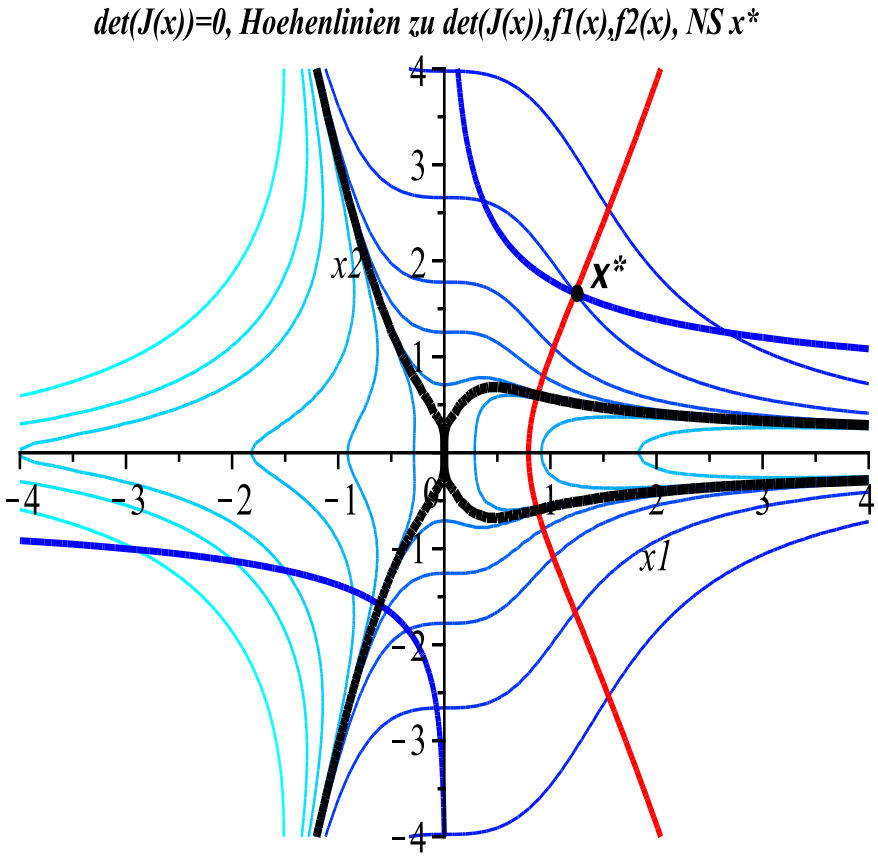


Abb. 2.4 Höhenlinienbild der Funktion $\det(\mathcal{J}(x))$,
Nullniveau $\det(\mathcal{J}(x)) = 0$ dick hervorgehoben,
dazu noch die Höhenlinien $f_{1,2}(x) = 0$ und Lösung x^*

Den Einzugsbereich einer Nullstelle einer Funktion überprüft man durch die Auflösung der Kontraktionsbedingung. Im skalaren Fall für das Newton-Verfahren heißt dies die Untersuchung der Bedingung $|g'(x)| < 1$ auf der Basis der zugehörigen Fixpunktgleichung

$$x = g(x) = x - \frac{f(x)}{f'(x)}, \quad g'(x) = \frac{f(x)\, f''(x)}{[f'(x)]^2}.$$

Der Einzugsbereich der Nullstelle ist dann ein Intervall, dort wo die Nullstelle anziehend ist, d. h. eine Kontraktion des Lösungsfehlers vorliegt.
Jedoch können auch Startwerte außerhalb des Intervalls konvergente Iterationsfolgen erzeugen. So ist z. B. für die Funktion $f(x) = e^x - 1$ mit $f'(x) \neq 0$ und der einzigen Nullstelle $x^* = 0$ die Bedingung $|g'(x)| = |1 - e^{-x}| < 1$ im Intervall $G = (-\ln(2), \infty)$ erfüllt. Aber jeder Startwert $x_0 < -\ln(2)$ liefert als nächste Iterierte $x_1 = x_0 - 1 + e^{-x_0} > 0$, so dass diese dann ins Intervall G fällt und eine konvergente Folge bewirkt.
Während im Eindimensionalen die Situation i. Allg. überschaubar bleibt, sind ähnliche Überlegungen im $\mathbb{R}^n,\ n \geq 2$, meist viel komplexer und schwieriger.

Maple-Prozedur
zum Newton-Verfahren für Gleichungssysteme der Dimension 2
Die Funktion ist als Liste von 2 Ausdrücken in den Variablen `x1` und `x2` definiert. Sie enthält die exakte Berechnung der Jacobi-Matrix, geeignete Abbruchkriterien, wahlweise Ausgabe von Zwischenergebnissen sowie die globale Größe `xv` für die Speicherung aller iterierten Vektoren. Damit können u. a. im Nachhinein die Iterationsverläufe graphisch dargestellt werden.

```
> Newton_System2 := proc(f::list,xstart::list,
                         maxiter::posint,etol::numeric,aus::name)
   local k,fh1,fh2,xx,xk,fxk,Jf,A,dx,Digits_old;
   global xv;

  Digits_old:=Digits;
  Digits:=32;
  xx:=matrix(2,0,[]);
  xk:=evalm(xstart):
  fxk:=evalf(eval(f,{x1=xk[1],x2=xk[2]}));
  xv:=evalm(concat(xx,xk));
  if aus=ja then
   fprintf(default,`\nIterationsverlauf\n`):
   fprintf(default,
          `k = %2.0f  x = [%+.16e, %+.16e]  f(x) = [%+.3e, %+.3e]\n`,
           0,xk[1],xk[2],fxk[1],fxk[2]):
  end if;
  if norm(fxk,2)<etol then RETURN(xk,0); end if;
  Jf:=array([[diff(f[1],x1),diff(f[1],x2)],
             [diff(f[2],x1),diff(f[2],x2)]]);
  for k from 1 by 1 to maxiter do
   A:=evalf(eval(Jf,{x1=xk[1],x2=xk[2]}));
```

```
 if det(A)=0 then
  lprint(‘Jacobi-Matrix singulaer‘):
  RETURN(xk,k);
 end if;
 dx:=linsolve(A,fxk);
 xk:=evalm(xk-dx);
 fxk:=evalf(eval(f,{x1=xk[1],x2=xk[2]}));
 xv:=evalm(concat(xv,xk));
 if aus=ja then
  fprintf(default,
          ‘k = %2.0f  x = [%+.16e, %+.16e]  f(x) = [%+.3e, %+.3e]\n‘,
           k,xk[1],xk[2],fxk[1],fxk[2]):
 end if;
 if norm(fxk,2)<etol then RETURN(xk,k); end if;
end do:
if k>maxiter then k:=k-1; end if;
[xk,k];
end proc:
```

Von Interesse sind noch einige Iterationsverläufe zu gegebenen Startvektoren.
Wir teilen die Iterationsverläufe in Grupppen (Fallbeispiele) ein:
- Startvektoren aus der näheren Umgebung von x^* und kurze Folgen,
- Startvektoren aus der weiteren Umgebung von x^* und längere Folgen,
- spezielle Startvektoren aus der weiteren Umgebung von x^* und sehr lange Folgen.

Zu jeder Gruppe werden wir eine Graphik erzeugen.

Die Ergebnisse werden unter verschiedenen Variablen abgelegt.
Man beachte in den abgekürzten Listung der Iterationsfolgen “auffällige“ Iterierte, die Ausdehnung des Gebietes der Iterierten sowie ihr manchmal chaotisches Verhalten (Zickzack-Verlauf mit nahen und fernen Punkten).

```
> # Startvektoren
  # Gruppe 1
  x0:=[ 4.0, 3.0];                  # 8 Iterationen
  #x0:=[ 1.4, 0.8];                 # 8 Iterationen
  #x0:=[ 0.1, 3.0];                 # 8 Iterationen
  #x0:=[ 1.0, 4.0];                 # 7 Iterationen
  # Gruppe 2
  #x0:=[ 3.0, -1.0];                # 21 Iterationen
  #x0:=[-1.0,-1.0];                 # 24 Iterationen
  #x0:=[1.0,0.4];                   # 22 Iterationen
  # Gruppe 3
  #x0:=[1,sqrt((sqrt(93.0)-9)/2)];  # 162 Iterationen
  #                                 # Jacobi-Matrix singulaer bei x0
  #x0:=[-0.1, 0.0];                 # 125 Iterationen
  #x0:=[ 0.0, 0.0];                 # Jacobi-Matrix singulaer bei x0

  erg:=Newton_System2(f,x0,200,1e-16,ja);
  evalm(erg[1]);
  erg[2];
  xv2:=evalm(xv);
  cd2:=coldim(xv2);                 # Variablenname xv2,cd2,xv3,cd3,...
```

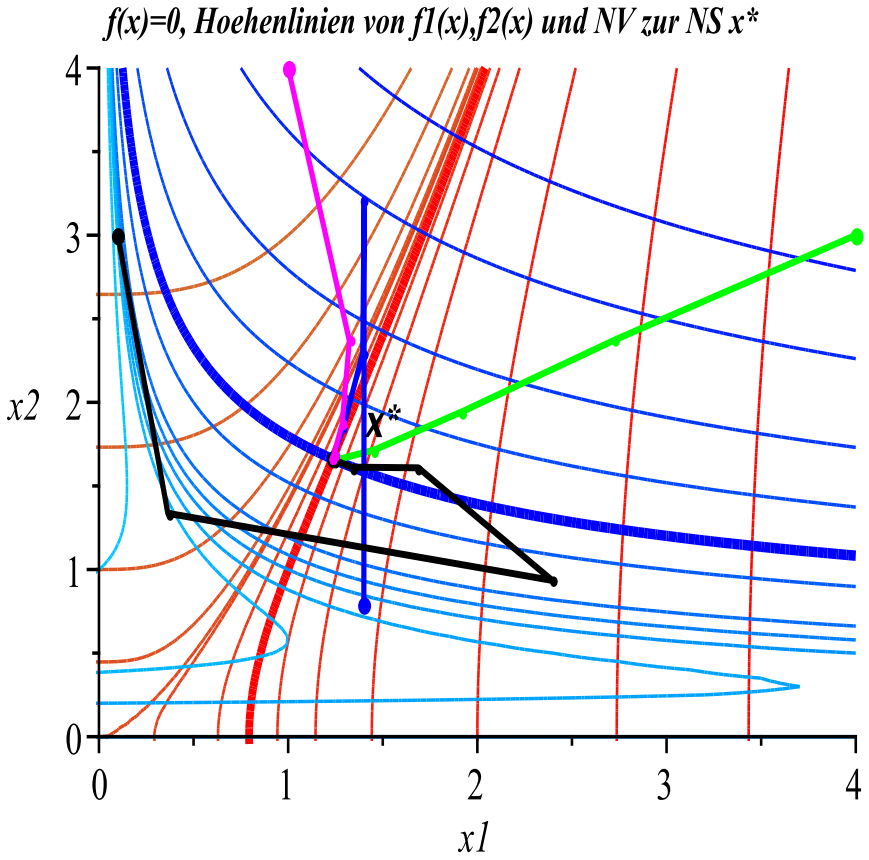


Abb. 2.5 Gruppe 1: Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit x^*, 4 konvergente Iterationsverläufe im 1. Quadranten

```
Iterationsverlauf
k =   0  x = [+4.0000000000000000e+00, +3.0000000000000000e+00]  f(x) = [+1.180e+02, +1.010e+02]
k =   1  x = [+2.7318382211999233e+00, +2.3760782058654399e+00]  f(x) = [+3.413e+01, +3.027e+01]
k =   2  x = [+1.9240799759255866e+00, +1.9467644654747724e+00]  f(x) = [+9.456e+00, +8.249e+00]
k =   3  x = [+1.4579710085385401e+00, +1.7163493140267236e+00]  f(x) = [+2.253e+00, +1.655e+00]
k =   4  x = [+1.2658699211729951e+00, +1.6587943265987405e+00]  f(x) = [+3.053e-01, +1.191e-01]
k =   5  x = [+1.2349232718235465e+00, +1.6611425320043797e+00]  f(x) = [+7.209e-03, -5.660e-04]
k =   6  x = [+1.2342747206359708e+00, +1.6615262239841493e+00]  f(x) = [+2.969e-06, -1.154e-06]
k =   7  x = [+1.2342744841145051e+00, +1.6615264667959071e+00]  f(x) = [+3.553e-13, -1.129e-13]
k =   8  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [+5.579e-27, -2.057e-27]
```

$$erg := xk, 8$$

$$[1.2342744841144759941 \quad 1.6615264667959338893]$$

$$8$$

$$xv2 := [[4.0,\ 2.7318382211999233276,\ 1.9240799759255866500,\ 1.4579710085385401046,\ 1.2658699211729950721,\ 1.2349232718235465279,\ 1.2342747206359707816,\ 1.2342744841145051459,\ 1.2342744841144759941],$$
$$[3.0,\ 2.3760782058654399080,\ 1.9467644654747724413,\ 1.7163493140267236198,\ 1.6587943265987404969,\ 1.6611425320043797070,\ 1.6615262239841492658,\ 1.6615264667959071467,\ 1.6615264667959338893]]$$

$$cd2 := 9$$

```
Iterationsverlauf
k =   0  x = [+1.4000000000000000e+00, +8.0000000000000000e-01]  f(x) = [+3.848e+00, -4.083e+00]
k =   1  x = [+1.4018236987955538e+00, +3.2184041861473202e+00]  f(x) = [-5.849e+00, +3.951e+01]
k =   2  x = [+1.3942598626362678e+00, +2.2959231799279307e+00]  f(x) = [-8.505e-01, +1.058e+01]
k =   3  x = [+1.2923893734337798e+00, +1.8519440065954482e+00]  f(x) = [-1.124e-01, +2.357e+00]
```

```
k =   4  x = [+1.2423311021971680e+00, +1.6861499794286957e+00]  f(x) = [-8.307e-03, +2.695e-01]
k =   5  x = [+1.2344384800469453e+00, +1.6620134301387784e+00]  f(x) = [-1.192e-04, +5.245e-03]
k =   6  x = [+1.2342745513522181e+00, +1.6615266631541427e+00]  f(x) = [-3.792e-08, +2.119e-06]
k =   7  x = [+1.2342744841144871e+00, +1.6615264667959659e+00]  f(x) = [-5.076e-15, +3.466e-13]
k =   8  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [-1.150e-28, +9.270e-27]
-------------------------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+1.0000000000000000e-01, +3.0000000000000000e+00]  f(x) = [-9.998e+00, -4.300e+00]
k =   1  x = [+3.6401031449334370e-01, +1.3363067698116001e+00]  f(x) = [-2.689e+00, -4.468e+00]
k =   2  x = [+2.3961957218258383e+00, +9.3459500192684459e-01]  f(x) = [+2.564e+01, -2.978e+00]
k =   3  x = [+1.6883965629715112e+00, +1.6082627282833947e+00]  f(x) = [+6.040e+00, +1.415e+00]
k =   4  x = [+1.3360696008961900e+00, +1.6124358057190179e+00]  f(x) = [+1.170e+00, -1.130e-02]
k =   5  x = [+1.2400528934627208e+00, +1.6563606988903876e+00]  f(x) = [+7.021e-02, -2.122e-02]
k =   6  x = [+1.2342921529019837e+00, +1.6615088552357575e+00]  f(x) = [+2.200e-04, -8.137e-05]
k =   7  x = [+1.2342744842776083e+00, +1.6615264666422709e+00]  f(x) = [+2.002e-09, -6.688e-10]
k =   8  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [+1.735e-19, -6.234e-20]
-------------------------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+1.0000000000000000e+00, +4.0000000000000000e+00]  f(x) = [-1.500e+01, +5.600e+01]
k =   1  x = [+1.3236775818639798e+00, +2.3677581863979849e+00]  f(x) = [-1.968e+00, +1.120e+01]
k =   2  x = [+1.2845275611809884e+00, +1.8653077098284843e+00]  f(x) = [-2.404e-01, +2.471e+00]
k =   3  x = [+1.2421146681953350e+00, +1.6883144049527245e+00]  f(x) = [-1.762e-02, +2.892e-01]
k =   4  x = [+1.2344513389091296e+00, +1.6620884513765596e+00]  f(x) = [-2.510e-04, +5.997e-03]
k =   5  x = [+1.2342745686181310e+00, +1.6615267245466893e+00]  f(x) = [-8.411e-08, +2.765e-06]
k =   6  x = [+1.2342744841144944e+00, +1.6615264667959886e+00]  f(x) = [-1.355e-14, +5.891e-13]
k =   7  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [-4.839e-28, +2.677e-26]
```

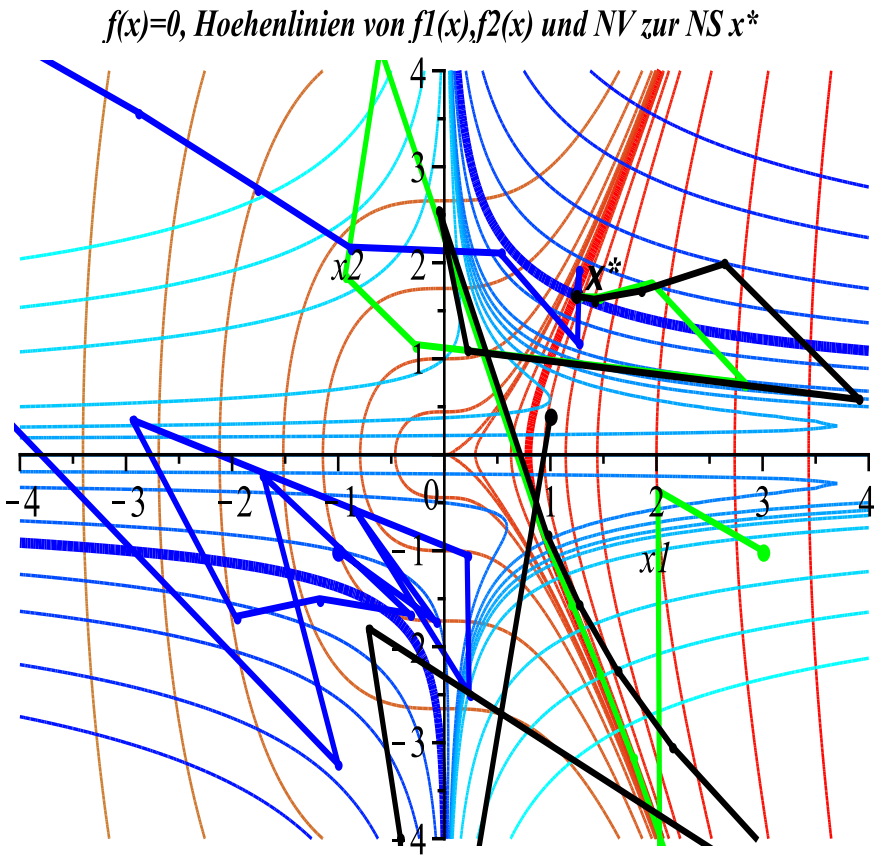


Abb. 2.6 Gruppe 2: Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit x^*, 3 Iterationsverläufe (einige Iterierte außerhalb des `view`-Gebiets)

```
Iterationsverlauf
k =   0  x = [+3.0000000000000000e+00, -1.0000000000000000e+00]  f(x) = [+5.200e+01, -6.000e+00]
k =   1  x = [+2.0138248847926267e+00, -3.7327188940092166e-01]  f(x) = [+1.519e+01, -3.731e+00]
k =   2  x = [+2.1139034516287187e+00, -2.3988725943808846e+01]  f(x) = [-5.576e+02, -2.916e+04]
k =   3  x = [+2.9494643912235231e+00, -1.2834236481526519e+01]  f(x) = [-1.144e+02, -6.226e+03]
k =   4  x = [+3.0016638615873737e+00, -8.4835206343021578e+00]  f(x) = [-1.888e+01, -1.828e+03]
k =   5  x = [+2.5869812207361014e+00, -6.0495115511159354e+00]  f(x) = [-2.970e+00, -5.707e+02]
k =   6  x = [+2.1551653242605633e+00, -4.3709060746851821e+00]  f(x) = [-8.447e-02, -1.796e+02]
k =   7  x = [+1.7788451469038839e+00, -3.1615554483586147e+00]  f(x) = [+2.621e-01, -5.705e+01]
k =   8  x = [+1.4651037650581349e+00, -2.2609732295257139e+00]  f(x) = [+1.778e-01, -1.867e+01]
k =   9  x = [+1.1966629222996589e+00, -1.5357278508869306e+00]  f(x) = [+6.879e-02, -6.799e+00]
k =  10  x = [+9.1245680427338362e-01, -7.6309261212580563e-01]  f(x) = [-6.293e-02, -3.642e+00]
k =  11  x = [-6.0217346052000273e-01, +4.2357830785717986e+00]  f(x) = [-1.938e+01, -5.400e+01]
k =  12  x = [-9.3488735898324883e-01, +1.8628513585118972e+00]  f(x) = [-6.104e+00, -1.191e+01]
k =  13  x = [-2.7779372881741935e-01, +1.1492760417102241e+00]  f(x) = [-2.364e+00, -5.571e+00]
k =  14  x = [+2.8374421904532954e+00, +7.4845621295854499e-01]  f(x) = [+4.413e+01, -3.559e+00]
k =  15  x = [+1.9562273671233085e+00, +1.7908515254659353e+00]  f(x) = [+1.077e+01, +5.445e+00]
k =  16  x = [+1.4644475682288503e+00, +1.6438263613186498e+00]  f(x) = [+2.579e+00, +8.611e-01]
k =  17  x = [+1.2646317583234485e+00, +1.6462612427302923e+00]  f(x) = [+3.349e-01, -3.907e-03]
k =  18  x = [+1.2347998367124693e+00, +1.6610215353879273e+00]  f(x) = [+6.482e-03, -2.247e-03]
k =  19  x = [+1.2342746288911975e+00, +1.6615263267991452e+00]  f(x) = [+1.789e-06, -6.270e-07]
k =  20  x = [+1.2342744841144870e+00, +1.6615264667959233e+00]  f(x) = [+1.356e-13, -4.728e-14]
k =  21  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [+7.820e-28, -2.735e-28]
-----------------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [-1.0000000000000000e+00, -1.0000000000000000e+00]  f(x) = [-4.000e+00, -2.000e+00]
k =   1  x = [-9.0909090909090909e-02, -1.7272727272727273e+00]  f(x) = [-3.985e+00, -1.804e+00]
k =   2  x = [-8.5085146902843036e-01, -5.6281944389824350e-01]  f(x) = [-2.549e+00, -3.285e+00]
k =   3  x = [+2.3440165521667788e-01, -2.4864309152525410e+00]  f(x) = [-7.157e+00, -5.117e+00]
k =   4  x = [+2.1520351268157176e-01, -1.0460310208344110e+00]  f(x) = [-2.074e+00, -3.200e+00]
k =   5  x = [-2.9428248691844088e+00, +3.6491459623129901e-01]  f(x) = [-5.210e+01, -4.508e+00]
k =   6  x = [-1.9688725850777464e+00, -1.6853430270302984e+00]  f(x) = [-1.910e+01, +7.110e+00]
k =   7  x = [-1.1744282422638954e+00, -1.4992969032945098e+00]  f(x) = [-6.488e+00, +1.457e+00]
k =   8  x = [-3.3473584958699687e-01, -1.6531717377806959e+00]  f(x) = [-3.808e+00, -8.345e-01]
k =   9  x = [-1.7048360030092231e+00, -2.2286285497715960e-01]  f(x) = [-1.096e+01, -3.758e+00]
k =  10  x = [-9.9960612457962570e-01, -3.2260466016018505e+00]  f(x) = [-1.341e+01, +3.279e+01]
k =  11  x = [-1.0362846477679965e+01, +7.5518835359786500e+00]  f(x) = [-2.284e+03, -4.475e+03]
k =  12  x = [-6.8576799839960523e+00, +5.8804750643511658e+00]  f(x) = [-6.806e+02, -1.404e+03]
k =  13  x = [-4.4998036319699559e+00, +4.5822144501078215e+00]  f(x) = [-2.042e+02, -4.415e+02]
k =  14  x = [-2.8949516954087857e+00, +3.5728401970479314e+00]  f(x) = [-6.229e+01, -1.396e+02]
k =  15  x = [-1.7682923103490635e+00, +2.7841930286248132e+00]  f(x) = [-1.981e+01, -4.495e+01]
k =  16  x = [-8.9652242961246063e-01, +2.1637752879111268e+00]  f(x) = [-7.123e+00, -1.525e+01]
k =  17  x = [+5.2417390055758209e-01, +2.1009779523691899e+00]  f(x) = [-5.126e+00, -1.240e+00]
k =  18  x = [+1.2555668497150631e+00, +1.1680008805036703e+00]  f(x) = [+1.594e+00, -3.167e+00]
k =  19  x = [+1.2742317796903546e+00, +1.9261303131902343e+00]  f(x) = [-5.721e-01, +3.179e+00]
k =  20  x = [+1.2440543929830180e+00, +1.7012984737145703e+00]  f(x) = [-4.364e-02, +4.248e-01]
k =  21  x = [+1.2346141415024374e+00, +1.6627090381885376e+00]  f(x) = [-8.256e-04, +1.248e-02]
k =  22  x = [+1.2342748316235025e+00, +1.6615275860957623e+00]  f(x) = [-5.431e-07, +1.192e-05]
k =  23  x = [+1.2342744841148076e+00, +1.6615264667969540e+00]  f(x) = [-3.585e-13, +1.093e-11]
k =  24  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [-2.261e-25, +9.205e-24]
-----------------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+1.0000000000000000e+00, +4.0000000000000000e-01]  f(x) = [+8.400e-01, -4.336e+00]
k =   1  x = [-2.7267987486965589e-01, -8.0950990615224192e+00]  f(x) = [-6.657e+01, +1.487e+02]
k =   2  x = [-4.1595248375308354e-01, -3.9793312507132711e+00]  f(x) = [-1.698e+01, +2.619e+01]
k =   3  x = [-7.1608361228956360e-01, -1.8067832594245163e+00]  f(x) = [-4.999e+00, +2.030e+00]
k =   4  x = [+9.0815088127187252e+00, -8.7652714282638681e+00]  f(x) = [+1.420e+03, -6.111e+03]
...
k =  16  x = [+1.8524422654916119e+00, +1.7062219951502087e+00]  f(x) = [+8.802e+00, +3.495e+00]
k =  17  x = [+1.4107181006270422e+00, +1.6205073568062878e+00]  f(x) = [+1.989e+00, +3.828e-01]
k =  18  x = [+1.2520005725521135e+00, +1.6494369844514813e+00]  f(x) = [+2.044e-01, -3.105e-02]
k =  19  x = [+1.2344468039888012e+00, +1.6613499189668765e+00]  f(x) = [+2.162e-03, -8.378e-04]
k =  20  x = [+1.2342744995914098e+00, +1.6615264525762746e+00]  f(x) = [+1.887e-07, -6.015e-08]
k =  21  x = [+1.2342744841144761e+00, +1.6615264667959338e+00]  f(x) = [+1.572e-15, -5.787e-16]
k =  22  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [+0.000e+00, +5.000e-31]
```

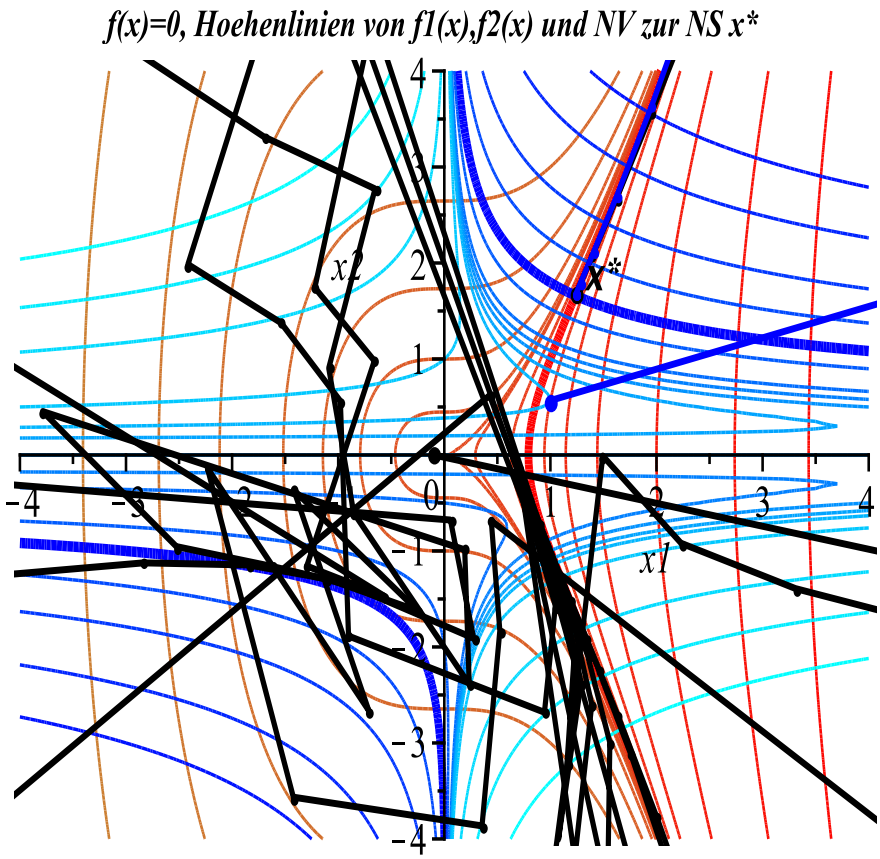


Abb. 2.7 Gruppe 3: Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit x^*, 2 Iterationsverläufe (Folge mit $x^{(0)} = \left(1, \sqrt{\frac{\sqrt{93}-9}{2}}\right)^T$ im 1. Quadranten)

```
Iterationsverlauf
k =   0  x = [-1.0000000000000000e-01, +0.0000000000000000e+00]  f(x) = [-1.002e+00, -4.000e+00]
k =   1  x = [+1.6600000000000000e+01, -4.0000000000000000e+00]  f(x) = [+9.132e+03, -1.062e+03]
k =   2  x = [+1.1072640023928235e+01, -3.1095137452482948e+00]  f(x) = [+2.704e+03, -3.338e+02]
k =   3  x = [+7.3903731044660168e+00, -2.4127626313895318e+00]  f(x) = [+8.005e+02, -1.054e+02]
k =   4  x = [+4.9395713236336345e+00, -1.8586247526777785e+00]  f(x) = [+2.366e+02, -3.386e+01]
k =   5  x = [+3.3116350266899093e+00, -1.3923256565930403e+00]  f(x) = [+6.970e+01, -1.155e+01]
k =   6  x = [+2.2324061104361382e+00, -9.1951957753780382e-01]  f(x) = [+2.041e+01, -4.816e+00]
...
k =  10  x = [+1.1386488382088621e+00, -1.5621836761669247e+00]  f(x) = [-4.879e-01, -6.779e+00]
...
k =  17  x = [-2.3859988751232830e+01, +1.4482901965555603e+01]  f(x) = [-2.738e+04, -7.250e+04]
...
k =  40  x = [-1.8441256166597472e+00, -1.1458937693256555e+00]  f(x) = [-1.486e+01, -7.936e-02]
...
k =  50  x = [+8.7214816925875657e+02, -7.4874308473819944e+02]  f(x) = [+1.326e+09, -3.661e+11]
...
k =  77  x = [+9.8182911848968222e-01, -1.0238760420933568e+00]  f(x) = [-1.554e-01, -4.030e+00]
k =  78  x = [+4.2640570013613680e-01, +6.2080977333202747e-01]  f(x) = [-1.230e+00, -4.519e+00]
k =  79  x = [-1.9052679604552831e+01, -1.7485124282694966e+01]  f(x) = [-1.414e+04, +1.019e+05]
...
k =  93  x = [+1.1136319536864108e+00, -1.3634238660740724e+00]  f(x) = [-9.673e-02, -5.459e+00]
...
k = 109  x = [+8.2627504721537566e-01, -7.3544661101260321e-01]  f(x) = [-4.126e-01, -3.593e+00]
...
```

```
k = 118  x = [+1.9542796332919829e+00, +3.5674594129760009e+00]  f(x) = [+1.201e+00, +8.116e+01]
k = 119  x = [+1.6223408882734564e+00, +2.6696744533550413e+00]  f(x) = [+4.128e-01, +2.420e+01]
k = 120  x = [+1.3964650815170455e+00, +2.0789265886800603e+00]  f(x) = [+1.246e-01, +6.468e+00]
k = 121  x = [+1.2742589299746052e+00, +1.7649911117558225e+00]  f(x) = [+2.293e-02, +1.241e+00]
k = 122  x = [+1.2374075706675488e+00, +1.6697795504846379e+00]  f(x) = [+1.218e-03, +9.112e-02]
k = 123  x = [+1.2342958779645379e+00, +1.6615839119590395e+00]  f(x) = [+4.659e-06, +6.279e-04]
k = 124  x = [+1.2342744851402958e+00, +1.6615264695906473e+00]  f(x) = [+8.963e-11, +3.048e-08]
k = 125  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [-1.741e-20, +7.180e-17]
-----------------------------------------------------------------------------------------------
Iterationsverlauf  (Achtung: bei Digits=10 anderer Verlauf)
k =   0  x = [+1.0000000000000000e+00, +5.6729655427869249e-01]  f(x) = [+6.782e-01, -4.385e+00]
k =   1  x = [+1.4287528353265292e+19, +7.5555870622720233e+19]  f(x) = [+5.833e+57, +6.163e+78]
k =   2  x = [+9.5250189021768614e+18, +5.8765677151004626e+19]  f(x) = [+1.728e+57, +1.933e+78]
k =   3  x = [+6.3500126014512410e+18, +4.5706637784114709e+19]  f(x) = [+5.121e+56, +6.063e+77]
...
k = 100  x = [+1.6008895942552207e+05, +8.7800580332711781e+07]  f(x) = [+4.967e+14, +1.084e+29]
...
k = 150  x = [+4.4180084312474590e+00, +1.2696589730973235e+01]  f(x) = [+1.027e+01, +9.026e+03]
k = 151  x = [+3.5867162558155760e+00, +9.2669323675736048e+00]  f(x) = [+5.407e+00, +2.841e+03]
k = 152  x = [+2.9163761058196832e+00, +6.7669220778472807e+00]  f(x) = [+2.818e+00, +8.929e+02]
k = 153  x = [+2.3790844904593054e+00, +4.9491815664302611e+00]  f(x) = [+1.437e+00, +2.795e+02]
k = 154  x = [+1.9544770321821392e+00, +3.6375772361371620e+00]  f(x) = [+7.002e-01, +8.644e+01]
k = 155  x = [+1.6303442142449851e+00, +2.7126585698647879e+00]  f(x) = [+3.085e-01, +2.583e+01]
k = 156  x = [+1.4038294939599356e+00, +2.1036580222507630e+00]  f(x) = [+1.078e-01, +6.965e+00]
k = 157  x = [+1.2778565736667790e+00, +1.7752358000725454e+00]  f(x) = [+2.181e-02, +1.374e+00]
k = 158  x = [+1.2379957552506238e+00, +1.6713819433031731e+00]  f(x) = [+1.270e-03, +1.089e-01]
k = 159  x = [+1.2343047893575474e+00, +1.6616081515722018e+00]  f(x) = [+5.565e-06, +8.924e-04]
k = 160  x = [+1.2342744861811570e+00, +1.6615264724418262e+00]  f(x) = [+1.291e-10, +6.155e-08]
k = 161  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [-2.453e-19, +2.927e-16]
k = 162  x = [+1.2342744841144760e+00, +1.6615264667959339e+00]  f(x) = [+0.000e+00, +5.000e-31]
-----------------------------------------------------------------------------------------------
```

Beispiel 2.4 Nichtlineares Gleichungssystem

$$\begin{aligned} f_1(x_1, x_2) &= (x_1 - x_2)^2 &&= 0, \\ f_2(x_1, x_2) &= x_1 + x_2 - 2 &&= 0 \quad \text{(linearer Anteil).} \end{aligned}$$

Dazu berechnet man die Jacobi-Matrix

$$\mathcal{J}(x) = \begin{pmatrix} 2(x_1 - x_2) & -2(x_1 - x_2) \\ 1 & 1 \end{pmatrix},$$

die einzige reelle Lösung $x^* = (1, 1)^T$ sowie an dieser Stelle die Determinante der Jacobi-Matrix

$$\det(\mathcal{J}(x^*)) = \begin{vmatrix} 0 & 0 \\ 1 & 1 \end{vmatrix} = 0.$$

Damit ist die Lösung x^* nicht regulär. Im skalaren Fall wäre also $f'(x^*) = 0$.
Praktisch heißt dies, dass im Newton-Verfahren die Konvergenzordnung 2 nicht mehr zu halten ist und auf 1 fällt. Die Fehler $|x_i^* - x_i^{(k)}|$ halbieren sich in jedem Schritt, die Werte $f_1(x^{(k)})$ werden entsprechend geviertelt. In der Iterationsfolge wird wegen der Linearität von $f_2(x)$ spätestens nach einem Schritt die Bedingung $f_2(x^{(k)}) = 0$ erfüllt sein. Daneben tendiert $f_1(x^{(k)})$ langsam gegen Null.
Weiterhin ist ein Startvektor mit $x_1 = x_2$ nicht geeignet.

k	$x_1^{(k)}$	$x_2^{(k)}$	$\|f(x^{(k)})\|_\infty = f_1(x^{(k)}),\ k \geq 1$
0	2	3	$f(x^{(0)}) = (1,3)^T$
1	0.75	1.25	2.500e-01
2	0.875	1.125	6.250e-02
3	0.937 5	1.062 5	1.563e-02
4	0.968 75	1.031 25	3.906e-03
5	0.984 375	1.015 625	9.766e-04
6	0.992 187 5	1.007 812 5	2.441e-04
7	0.996 093 75	1.003 906 25	6.104e-05
8	0.998 046 875	1.001 953 125	1.526e-05
9	0.999 023 437 5	1.000 976 562 5	3.815e-06
10	0.999 511 718 75	1.000 488 281 25	9.537e-07
11	0.999 755 859 375	1.000 244 140 625	2.384e-07
12	0.999 877 929 687 5	1.000 122 070 312 5	5.960e-08
13	0.999 938 964 843 75	1.000 061 035 156 25	1.490e-08
14	0.999 969 482 421 875	1.000 030 517 578 125	3.725e-09
15	0.999 984 741 210 937 5	1.000 015 258 789 062 5	9.313e-10
16	0.999 992 370 605 468 75	1.000 007 629 394 531 2	2.328e-10
17	0.999 996 185 302 734 38	1.000 003 814 697 265 6	5.821e-11
18	0.999 998 092 651 367 19	1.000 001 907 348 632 8	1.455e-11
19	0.999 999 046 325 683 59	1.000 000 953 674 316 4	3.638e-12
20	0.999 999 523 162 841 80	1.000 000 476 837 158 2	9.095e-13
21	0.999 999 761 581 420 90	1.000 000 238 418 579 1	2.274e-13
22	0.999 999 880 790 710 45	1.000 000 119 209 289 6	5.684e-14
23	0.999 999 940 395 355 22	1.000 000 059 604 644 8	1.421e-14
24	0.999 999 970 197 677 61	1.000 000 029 802 322 4	3.553e-15
25	0.999 999 985 098 838 81	1.000 000 014 901 161 2	8.882e-16
26	0.999 999 992 549 419 40	1.000 000 007 450 580 6	2.220e-16
27	0.999 999 996 274 709 70	1.000 000 003 725 290 3	5.551e-17

Tab. 2.3 Iterationsverlauf des Newton-Verfahrens mit $x^{(k)}$ und $\|f(x^{(k)})\|_\infty = f_1(x^{(k)}),\ k = 1, 2, ..., 27$

```
> f:=[(x1-x2)^2, x1+x2-2];
> # 0-Kontur fuer f[1] wird so nicht angzeigt, deshalb  contours=[0.0005]
  #pl1:=contourplot(f[1],x1=-2..4,x2=-2..4,grid=[241,241],contours=[0]):
  pl1:=contourplot(f[1],x1=-2..4,x2=-2..4,grid=[241,241],color=red,
                   thickness=3,contours=[0.0005]):
  pl1a:=contourplot(f[1],x1=-2..4,x2=-2..4,grid=[61,61],
                   coloring=[gold,red],thickness=1,
                   contours=[-20,-10,-5,-2,-0.5,0.5,2,5,10,20]):
  pl2:=contourplot(f[2],x1=-2..4,x2=-2..4,grid=[61,61],color=blue,
```

```
                thickness=3,contours=[0]):
pl2a:=contourplot(f[2],x1=-2..4,x2=-2..4,grid=[61,61],
                coloring=[cyan,blue],thickness=1,contours=11):
pl3:=pointplot(xs,color=black,symbol=solidcircle,symbolsize=20):
pl3a:=textplot([1.3,1.0,`x*`],font=[HELVETICA,BOLD,10]):

> display(pl1,pl1a,pl2,pl2a,pl3,pl3a,view=[-2..4,-2..4],
                title=`f(x)=0, Hoehenlinien zu f1(x),f2(x), NS x*`,
                titlefont=[TIMES,BOLD,9]);
```

Die Höhenlinien der Koordinatenfunktionen $f_1(x)$ und $f_2(x)$ sind jeweils Büschel paralleler Geraden $x_1 - x_2 = c$ bzw. $x_1 + x_2 - 2 = c$, die zueinander orthogonal sind. Das trifft auch auf die Nullhöhenlinien zu. Der Einzugsbereich der Lösung x^* ist $\mathbb{R}^2 \backslash \{x^*\}$.

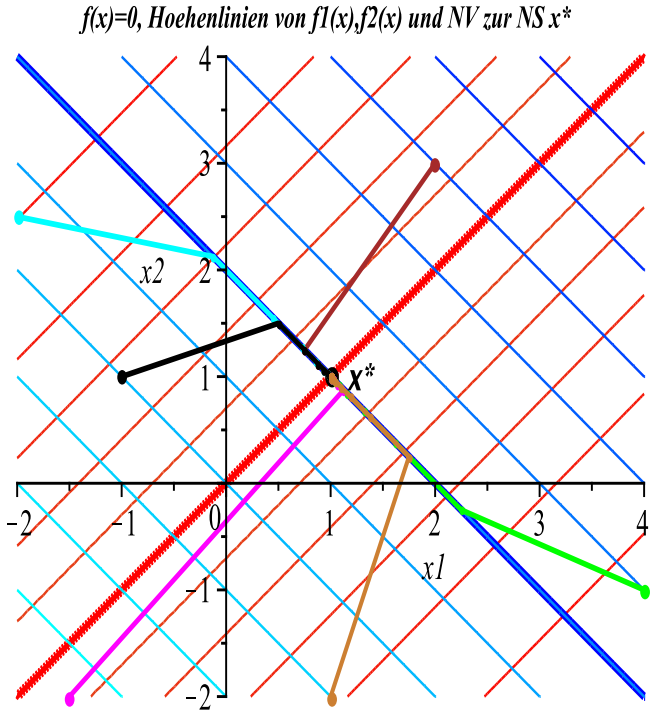


Abb. 2.8 Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit x^*, 6 konvergente Iterationsverläufe

```
> x0:=[  2.0,  3.0]:                  # 27 Iterationen
  #x0:=[  4.0, -1.0]:                 # 29 Iterationen
  #x0:=[ -2.0,  2.5]:                 # 29 Iterationen
  #x0:=[ -1.0,  1.0]:                 # 28 Iterationen
  #x0:=[ -1.5, -2.0]:                 # 29 Iterationen
  #x0:=[  1.0, -2.0]:                 # 29 Iterationen
  #x0:=[ c, c]:                       # Jacobi-Matrix singulaer bei x0

  erg:=Newton_System2(f,x0,50,1e-16,ja);
  evalm(erg[1]);
  erg[2];
  xv1:=evalm(xv);
  cd1:=coldim(xv1);                   # Variablenname xv1,cd1,xv2,cd2,...
```

Beispiel 2.5 Sei

$$\begin{aligned} f_1(x_1, x_2) &= \frac{1}{4}(x_1 \sin(x_2) + x_2) - x_1 &= 0, \\ f_2(x_1, x_2) &= \arctan\Big(\frac{4}{x_1 + x_2}\Big) - x_2 &= 0. \end{aligned}$$

Die zwei reellen Lösungen des Problems sind
$x_1^* = (\ 0.388\,899\,948\,773\,359,\quad 1.193\,984\,154\,564\,548)^T$ und
$x_2^* = (-0.246\,902\,265\,787\,981,\ -1.219\,426\,089\,292\,497)^T$.
Als Startvektoren wählen wir $x^{(0)} = (0, 0.5)^T,\ (0, -0.5)^T$. Der Vektor $x^{(0)} = (c, -c)^T$ führt zum Abbruch wegen Nulldivision.
Die Jacobi-Matrix ist

$$\mathcal{J}(x) = \begin{pmatrix} 1 - \frac{1}{4}\sin(x_2) & -\frac{1}{4}(x_1 \cos(x_2) + 1) \\ \frac{4}{16 + (x_1 + x_2)^2} & 1 + \frac{4}{16 + (x_1 + x_2)^2} \end{pmatrix}, \quad \det(\mathcal{J}(x_{1,2}^*)) \neq 0.$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$\|f(x^{(k)})\|_2$
0	0	0.5	9.547e-01
1	0.338 745 099 367 488 12	1.192 577 345 756 1602	4.013e-02
2	0.388 878 362 006 306 89	1.193 948 078 324 8653	4.894e-05
3	0.388 899 948 740 046 19	1.193 984 154 519 9209	6.280e-11
4	0.388 899 948 773 359 11	1.193 984 154 564 5484	1.216e-22
0	0	-0.5	9.547e-01
1	-0.269 296 920 797 123 60	-1.206 295 504 486 1088	3.257e-02
2	-0.246 877 570 739 366 73	-1.219 429 288 188 4876	3.126e-05
3	-0.246 902 265 781 213 15	-1.219 426 089 286 9775	1.086e-11
4	-0.246 902 265 787 980 98	-1.219 426 089 292 4968	3.558e-24

Tab. 2.4 Iterationsverlauf des Newton-Verfahrens mit $x^{(0)} = (0, \pm 0.5)^T$ sowie $x^{(k)}$ und $\|f(x^{(k)})\|_2,\ k = 0, 1, ..., 4$

Betrachten wir uns das Höhenlinienbild der Koordinatenfunktionen $f_1(x)$ und $f_2(x)$. Es zeigt auch, dass $x_{1,2}^*$ in der Tat die einzigen reellen Lösungen sind. Nahe der Geraden $x_2 = -x_1$ hat die Funktion $f_2(x)$ viele Niveauwerte.

```
> # Prozedur
  n:=2;
  x:=vector(n):
  f:=x->evalm([x[1]-1/4*(x[1]*sin(x[2])+x[2]),
               x[2]-arctan(4/(x[1]+x[2]))]);
```

```
> pl1:=contourplot(f(x)[1],x[1]=-2..2,x[2]=-2..2,grid=[41,41],color=red,
                   thickness=3,contours=[0]):
  pl1a:=contourplot(f(x)[1],x[1]=-2..2,x[2]=-2..2,grid=[41,41],
                   coloring=[gold,red],thickness=1,contours=10):
  pl2:=contourplot(f(x)[2],x[1]=-2..2,x[2]=-2..2,grid=[41,41],color=blue,
                   thickness=3,contours=[0]):
  pl2a:=contourplot(f(x)[2],x[1]=-2..2,x[2]=-2..2,grid=[41,41],
                   coloring=[cyan,blue],thickness=1,contours=10):
  pl3:=pointplot([xs1,xs2],color=black,symbol=solidcircle,symbolsize=16):
  pl4:=textplot([[0.75,1.3,`x1*`],[-0.5,-1.4,`x2*`]],
                   font=[HELVETICA,BOLD,10]):

> display(pl1,pl1a,pl2,pl2a,pl3,pl4,view=[-2..2,-2..2],
                   title=`f(x)=0, Hoehenlinien zu f1(x),f2(x), NS x1*,x2*`,
                   titlefont=[TIMES,BOLD,9]);
```

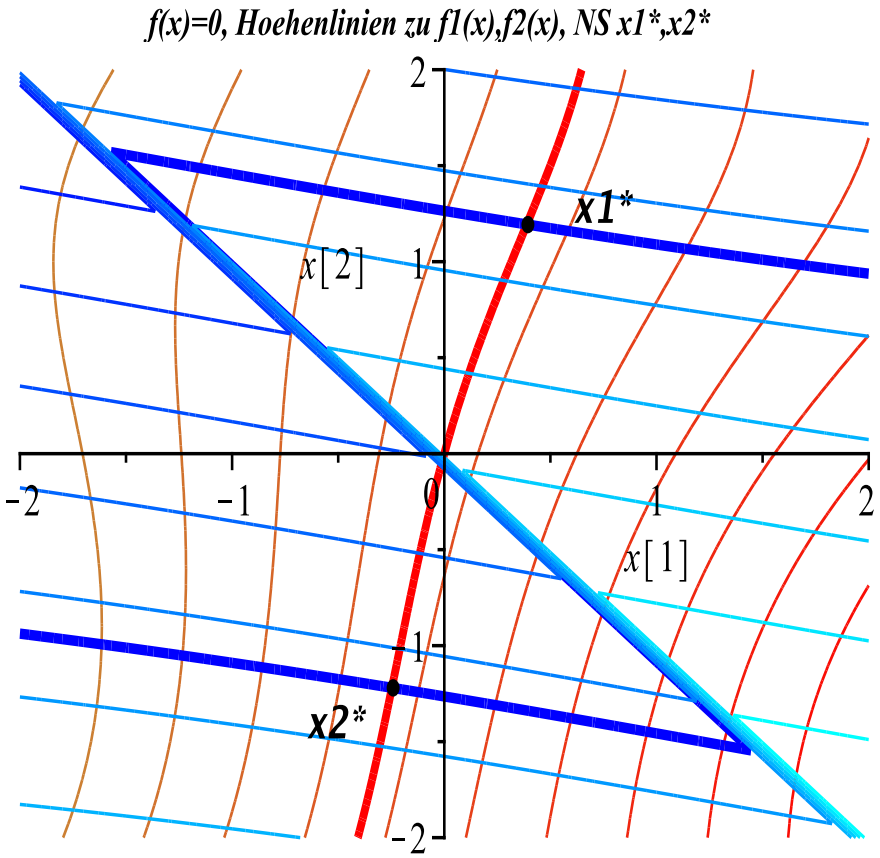


Abb. 2.9 Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit Lösungen $x^*_{1,2}$, Nullniveaus dick hervorgehoben

Die Jacobi-Matrix hat die Determinante

$$\det(\mathcal{J}(x)) = \frac{84 + (x_1 + x_2)^2[4 + \sin(x_2)] + 20\sin(x_2) + 4x_1\cos(x_2)}{4[(x_1 + x_2)^2 + 16]}.$$

Man kann nachprüfen, dass die Determinante nirgends verschwindet.

Auf der Geraden $x_2 = -x_1$ ist die Koordinatenfunktion $f_2(x)$ nicht definiert. Aber für jeden anderen Startvektor werden die Iterationsfolgen gegen eine der Nullstellen konvergieren.
Der Einzugsbereich der Nullstelle x_1^* ist somit $M_1 = \{(x_1, x_2)^T : x_2 > -x_1\}$, der von x_2^* ist $M_2 = \{(x_1, x_2)^T : x_2 < -x_1\}$.

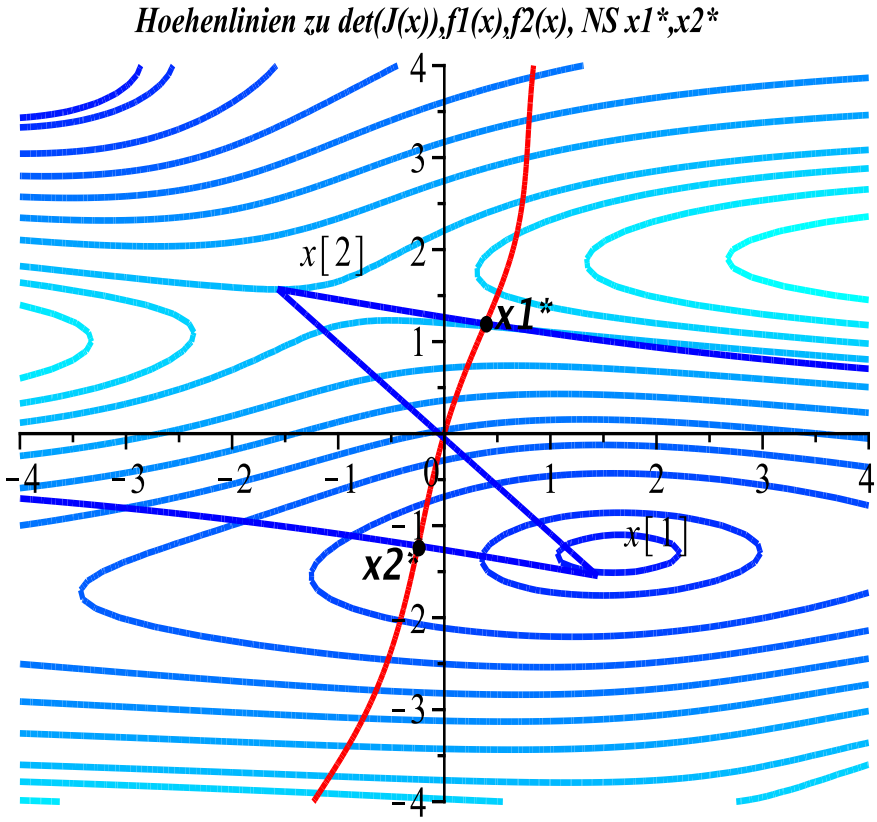


Abb. 2.10 Höhenlinienbild der Funktion $\det(\mathcal{J}(x))$ mit `contours=[0.85,0.9,0.95,1,1.1,1.2,1.3,1.4,1.5,1.6,1.63]`, dazu noch die Höhenlinien $f_{1,2}(x) = 0$ und Lösungen $x_{1,2}^*$

Maple-Prozedur
zum Newton-Verfahren für Gleichungssysteme der Dimension n
Die Funktion ist als Prozedur von n Ausdrücken in der Vektorvariablen `x[1..n]` definiert. Sie enthält die exakte Berechnung der Jacobi-Matrix, geeignete Abbruchkriterien, wahlweise Ausgabe von Zwischenergebnissen sowie die globale Größe `xv` für die Speicherung aller iterierten Vektoren. Damit können u. a. im Nachhinein die Iterationsverläufe graphisch dargestellt werden.
In unseren Beispielen ist $n = 2$.

```
> Newton_Systemn:=proc(f::procedure,n::posint,xstart::vector,
                       maxiter::posint,etol::numeric,aus::name)
   local k,xk,xx,i,j,v,A,df,df_hilf,fh,fh1,dx,fxk,Digits_old;
   global xv;
```

```
Digits_old := Digits;
Digits := 32;
fh1:=`%+.16e`;           # Ausgabeformate einstellen
fh:=fh1;
for k from 2 to n do
 fh:=cat(fh,`, `,fh1);
end do;

xx:=matrix(n,0,[]);
xk:=evalf(evalm(xstart)):
fxk:=evalm(f(xk)):
xv:=evalm(concat(xx,xk));
if aus=ja then
 fprintf(default,`\nIterationsverlauf\n`):
 fprintf(default,`k = %3.0f  x = [`||fh||`]  |f(x)| = %.3e\n`,
                  0,seq(xk[i],i=1..n),norm(fxk,2));
end if;
if norm(fxk,2)<etol then RETURN(xk,0); end if;
v:=vector(n):
df:=matrix(n,n);
A:=matrix(n,n):

# Jacobi-Matrix, Komponenten als Funktionen
for i from 1 to n do
 for j from 1 to n do
  df_hilf:=evalm(diff(f(v)[i],v[j]));
  df[i,j]:=unapply(df_hilf,v);
 end do;
end do;

for k from 1 by 1 to maxiter do
 for i from 1 to n do
  for j from 1 to n do
   A[i,j]:= evalm(df[i,j](xk));
  end do;
 end do;
  if det(A)=0 then
  lprint(`Jacobi-Matrix singulaer`): RETURN(xk,k);
 end if;
 dx:=linsolve(A,fxk);
 xk:=evalm(xk-dx);
 fxk:=evalf(evalm(f(xk)));
 xv:=evalm(concat(xv,xk));

 if aus=ja then
  printf(`k = %3.0f  x = [`||fh||`]  |f(x)| = %.3e\n`,
          k,seq(xk[i],i=1..n),norm(fxk,2));
 end if;
 if evalf(norm(evalm(fxk),2))<etol then RETURN(xk,k); end if;
end do:

if k>maxiter then k:=k-1; end if;
[xk,k];
end:
```

Von Interesse sind noch einige Iterationsverläufe zu gegebenen Startvektoren.
Wir teilen die Iterationsverläufe in 2 Grupppen für die Einzugsbreiche M_1 und M_2 ein.
Zu jeder Gruppe werden wir eine Graphik erzeugen. Auf eine tabellarische Ausgabe verzichten wir jedoch, da die Iterationsfolgen nur aus wenigen Iterierten bestehen.

```
> # Gruppe 1
  x0:=vector(n,[-1.5, 2.0]):        # 4-5 Iterationen
  #x0:=vector(n,[ 1.0, 2.0]):
  #x0:=vector(n,[ 2.0, 1.0]):
  #x0:=vector(n,[ 2.0,-1.0]):
  # Gruppe 2
  #x0:=vector(n,[ 1.0,-2.0]):       # 4-5 Iterationen
  #x0:=vector(n,[-1.5,-2.0]):
  #x0:=vector(n,[-2.0,-0.5]):
  #x0:=vector(n,[-2.0, 1.5]):

  erg:=Newton_Systemn(f,n,x0,20,1e-16,ja);
  evalm(erg[1]);
  erg[2];
  xv3:=evalm(xv);
  cd3:=coldim(xv3);                 # Variablenname xv3,cd3,xv4,cd4,...
```

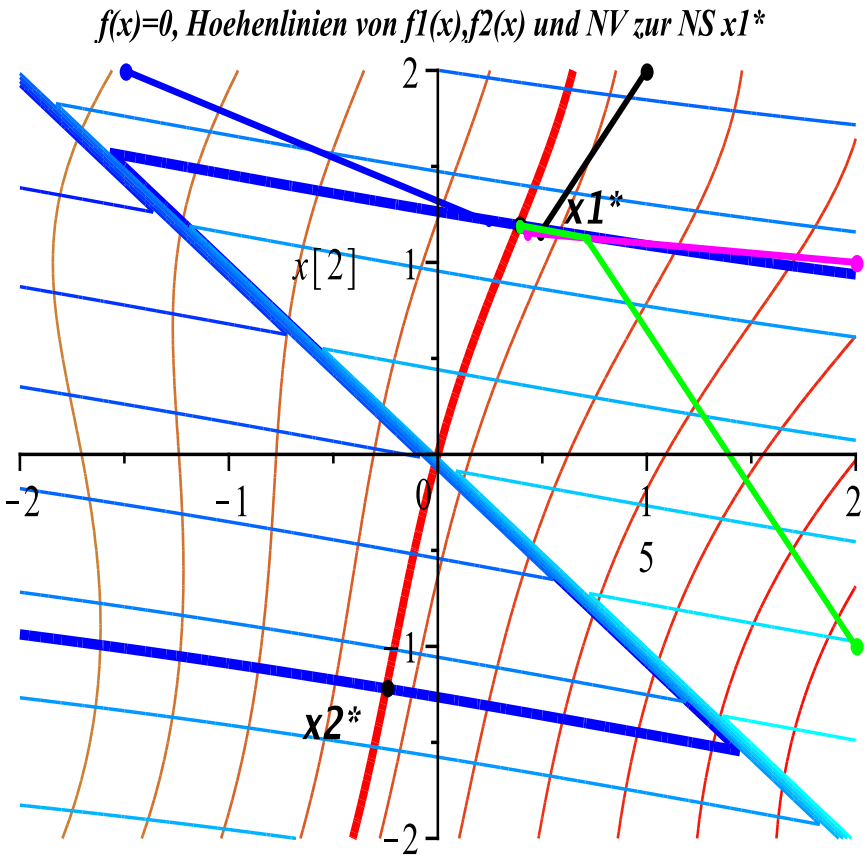


Abb. 2.11 Gruppe 1: Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit $x^*_{1,2}$, 4 konvergente Iterationsverläufe in M_1 zu x^*_1

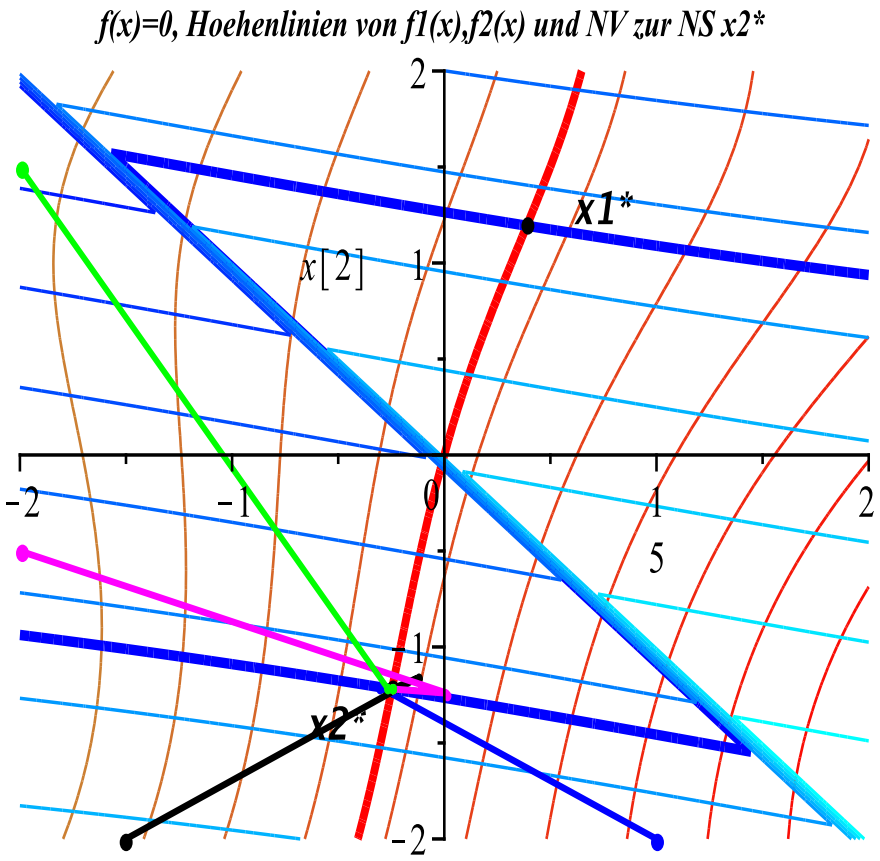


Abb. 2.12 Gruppe 2: Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit $x^*_{1,2}$, 4 konvergente Iterationsverläufe in M_2 zu x^*_2

Beispiel 2.6 Nichtlineares Gleichungssystem

$$\begin{aligned} f_1(x_1,x_2) &= 10(x_2 - x_1^2) &&= 0 \quad \text{(linear bez. } x_2\text{)}, \\ f_2(x_1,x_2) &= 1 - x_1 &&= 0 \quad \text{(linearer Anteil)}. \end{aligned}$$

Dazu berechnet man die reguläre Jacobi-Matrix

$$\mathcal{J}(x) = \begin{pmatrix} -20x_1 & 10 \\ -1 & 0 \end{pmatrix} = 10$$

und die einzige reelle Lösung $x^* = (1,1)^T$. Ihr Einzugsbereich ist die ganze reelle Ebene.
In der Iterationsfolge wird wegen der Linearität von $f_2(x)$ spätestens nach einem Schritt die Bedingung $f_2(x^{(1)}) = 0$ erfüllt sein. Dann haben wir wegen der Linearität von $f_1(x)$ bezüglich der Variablen x_2 im nächsten Schritt $f_1(x^{(2)}) = 0$ und die Iteration ist beeendet.

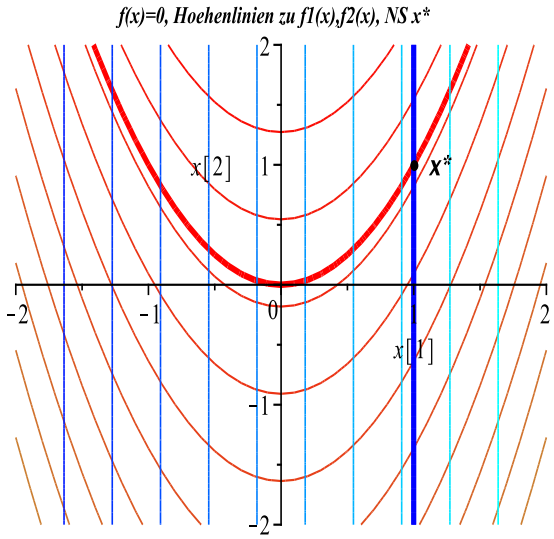


Abb. 2.13
Höhenlinienbild der Funktionen $f_1(x)$ und $f_2(x)$ mit x^*

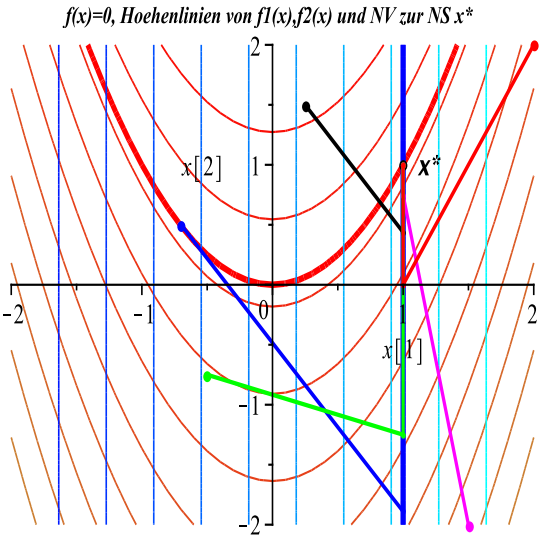


Abb. 2.14
Höhenlinienbild von $f_1(x)$ und $f_2(x)$, 5 konvergente Iterationsverläufe zu x^*

```
> x0:=vector(n,[-0.7,  0.5]):       # 2 Iterationen
  #x0:=vector(n,[ 0.25,1.5]):
  #x0:=vector(n,[ 1.5,-2.0]):
  #x0:=vector(n,[-0.5,-0.75]):
  #x0:=vector(n,[ 2.0, 2.0]):

  erg:=Newton_Systemn(f,n,x0,20,1e-16,ja);
  evalm(erg[1]);
  erg[2];
  xv3:=evalm(xv);
  cd3:=coldim(xv3);                 # Variablenname xv3,cd3,xv4,cd4,...
```

```
Iterationsverlauf
k =   0  x = [-7.0000000000000000e-01, +5.0000000000000000e-01]  |f(x)| = 1.703e+00
k =   1  x = [+1.0000000000000000e+00, -1.8900000000000000e+00]  |f(x)| = 2.890e+01
k =   2  x = [+1.0000000000000000e+00, +1.0000000000000000e+00]  |f(x)| = 0.000e+00
--------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+2.5000000000000000e-01, +1.5000000000000000e+00]  |f(x)| = 1.439e+01
k =   1  x = [+1.0000000000000000e+00, +4.3750000000000000e-01]  |f(x)| = 5.625e+00
k =   2  x = [+1.0000000000000000e+00, +1.0000000000000000e+00]  |f(x)| = 0.000e+00
--------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+1.5000000000000000e+00, -2.0000000000000000e+00]  |f(x)| = 4.250e+01
k =   1  x = [+1.0000000000000000e+00, +7.5000000000000000e-01]  |f(x)| = 2.500e+00
k =   2  x = [+1.0000000000000000e+00, +1.0000000000000000e+00]  |f(x)| = 0.000e+00
--------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [-5.0000000000000000e-01, -7.5000000000000000e-01]  |f(x)| = 1.011e+01
k =   1  x = [+1.0000000000000000e+00, -1.2500000000000000e+00]  |f(x)| = 2.250e+01
k =   2  x = [+1.0000000000000000e+00, +1.0000000000000000e+00]  |f(x)| = 0.000e+00
--------------------------------------------------------------------------------
Iterationsverlauf
k =   0  x = [+2.0000000000000000e+00, +2.0000000000000000e+00]  |f(x)| = 2.002e+01
k =   1  x = [+1.0000000000000000e+00, +0.0000000000000000e+00]  |f(x)| = 1.000e+01
k =   2  x = [+1.0000000000000000e+00, +1.0000000000000000e+00]  |f(x)| = 0.000e+00
```

Kapitel 3

Die Einheitswurzeln

3.1 Newton-Verfahren und Einzugsbereiche

Im Abschnitt 1.1 haben wir schon einige Betrachtungen zur Funktion $f(z) = z^m - 1$ (1.3) und ihrem Betrag gemacht.
Diese sollen hier in Maple und MATLAB weitergeführt werden. Dazu kommen Rechnungen mit dem Newton-Verfahren im Komplexen (Prozedur `Newton_komplex` aus Abschnitt 2.1) sowie Untersuchungen zu den Einzugsgebieten der verschiedenen Nullstellen mit zahlreichen graphischen Darstellungen. Da $z^m - 1$ ein komplexes Polynom ist, wird die Umsetzung des Newton-Verfahrens ohne größere Probleme möglich sein. Wie im reellen Fall $x^m - 1$ ist der Koordinatenursprung als Startpunkt nicht geeignet. Die Ergebnisse zu numerischen Rechnungen basieren hier auf $m = 5$.

$f(z) = z^m - 1,\ f'(z) = mz^{m-1}$, Nullstellen von $f(z)$

```
> Digits:=16:
> m:='m':
  f:=unapply(z^m-1,z); f(z);
  fs:=D(f); fs(z);
  simplify(%);
```

$$f := z \to z^m - 1$$
$$z^m - 1$$
$$fs := z \to \frac{z^m\, m}{z}$$
$$\frac{z^m\, m}{z}$$
$$z^{m-1}\, m$$

```
> s1:=solve(f(z));
  s1:=solve(f(z),z);
```

$$s1 := \{m = m, z = 1\},\ \{m = 0, z = z\}$$
$$s1 := 1$$

```
> m:=5:
  s1:=solve(f(z));    # analog  s1:=solve(f(z),z);
  evalf(s1);
```

$$s1 := 1,\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}}$$

$$1.,\ 0.3090169943749475 + 0.9510565162951535\,\mathrm{I},\ -0.8090169943749475 + 0.5877852522924732\,\mathrm{I},$$
$$-0.8090169943749475 - 0.5877852522924732\,\mathrm{I},\ 0.3090169943749475 - 0.9510565162951535\,\mathrm{I}$$

```
> fsolve(f(z),z,complex);
  fsolve(f(z),z,0.9-0.1*I..1.1+0.1*I);
  fsolve(f(z),z,0.9-0.1*I..1.1+0.1*I,complex);
  fsolve(f(z),z,0.2+0.8*I..0.4+1*I,complex);
  fsolve(f(z),z,-1+0.5*I..-0.7+0.6*I,complex);
```

$$-0.8090169943749475 - 0.5877852522924732\,\mathrm{I},\ -0.8090169943749475 + 0.5877852522924732\,\mathrm{I},$$
$$0.3090169943749475 - 0.9510565162951535\,\mathrm{I},\ 0.3090169943749475 + 0.9510565162951535\,\mathrm{I},\ 1.$$
$$1.$$
$$1.$$
$$0.3090169943749474 + 0.9510565162951536\,\mathrm{I}$$
$$-0.8090169943749474 + 0.5877852522924731\,\mathrm{I}$$

$|f(z)|$, die Ableitung $|f(z)|'$ wird wegen der Betragsfunktion natürlich Schwierigkeiten bereiten.

```
> m:='m':
  bf:=unapply(abs(z^m-1),z);
  bf(z);
  bfs:=D(bf);          # Ableitung von bf
  bfs(z);
  simplify(%);
  zz:=1+1*I;
  bfs(zz);             # Problem, nicht auswertbar
```

$$bf := z \to |z^m - 1|$$
$$|z^m - 1|$$
$$bfs := z \to \frac{\mathrm{abs}(1, z^m - 1) z^m m}{z}$$
$$\frac{\mathrm{abs}(1, z^m - 1) z^m m}{z}$$
$$\mathrm{abs}(1, z^m - 1) z^{m-1} m$$
$$zz := 1 + \mathrm{I}$$
$$\left(\frac{1}{2} - \frac{1}{2}\mathrm{I}\right) \mathrm{abs}(1, (1+\mathrm{I})^m - 1)(1+\mathrm{I})^m m$$

```
> m:=5:
  bf:=unapply(abs(z^m-1),z);
  bf(z);
  bfs:=D(bf);          # Ableitung von bf
  bfs(z);
  simplify(%);
  zz:=1+1*I;
  bfs(zz);             # Problem, nicht auswertbar
```

$$bf := z \to |z^5 - 1|$$
$$|z^5 - 1|$$
$$bfs := z \to 5\,\mathrm{abs}(1, z^5 - 1) z^4$$
$$5\,\mathrm{abs}(1, z^5 - 1) z^4$$
$$zz := 1 + \mathrm{I}$$

```
Error, (in simpl/abs) abs is not differentiable at non-real arguments
```

Nullstellen von $|f(z)|$, Kommando `fsolve` anders als bei $f(z)$

```
> m:='m':
  bf:=unapply(abs(z^m-1),z):
  s2:=solve(bf(z));
  s2:=solve(bf(z),z);
```

$$s2 := \{m = m, z = 1\},\ \{m = 0, z = z\}$$
$$s2 := 1$$

```
> m:=5:
  s2:=solve(bf(z));    # analog  s2:=solve(bf(z),z);
  evalf(s2);
```

$$s2 := 1,\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}+\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}-\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5-\sqrt{5}},\ -\frac{1}{4}+\frac{1}{4}\sqrt{5}-\frac{1}{4}\mathrm{I}\sqrt{2}\sqrt{5+\sqrt{5}}$$
$$1.,\ 0.3090169943749475 + 0.9510565162951535\,\mathrm{I},\ -0.8090169943749475 + 0.5877852522924732\,\mathrm{I},$$
$$-0.8090169943749475 - 0.5877852522924732\,\mathrm{I},\ 0.3090169943749475 - 0.9510565162951535\,\mathrm{I}$$

```
> fsolve(bf(z),z,complex);
  fsolve(bf(z),z,0.9-0.1*I..1.1+0.1*I);
  fsolve(bf(z),z,0.9-0.1*I..1.1+0.1*I,complex);
  fsolve(bf(z),z,0.2+0.8*I..0.4+1*I,complex);
  fsolve(bf(z),z,-1+0.5*I..-0.7+0.6*I,complex);
```

$$1. + 0.\,\mathrm{I}$$
$$1.000000000000000$$
$$1.000000000000000 + 0.\,\mathrm{I}$$
$$fsolve(|z^5 - 1|, z, 0.2 + 0.8\,\mathrm{I}..0.4 + 1.\,\mathrm{I}, complex)$$
$$fsolve(|z^5 - 1|, z, -1. + 0.5\,\mathrm{I}.. - 0.7 + 0.6\,\mathrm{I}, complex)$$

Bei Rechnungen mit dem Newton-Verfahren im Komplexen verwenden wir $f(z)$ sowie die Prozedur

```
Newton_komplex := proc(f::procedure,zstart::complex,
                       maxiter::posint,aus::name)::complex;
```

Ihre Anwendung auf die Betragsfunktion $|f(z)|$ führt wegen des Auftretens der Ableitung zum Abbruch und zu einer Fehlermeldung.

```
> m:=5:
  bf:=unapply(abs(z^m-1),z):
  xsb:=Newton_komplex(bf,2+0.1*I,10,ja);

 Iterationsverlauf
 k =  0   z = +2.0000000000000000e+00 +1.0000000000000000e-01i,
             f(z) = +3.123e+01 +0.000e+00i, |f(z)|= +3.123e+01
 Error, (in simpl/abs) abs is not differentiable at non-real arguments
```

Für die graphischen Auswertungen verwenden wir den Real- und Imaginärteil $\Re(f(z))$ und $\Im(f(z))$ der Funktion $f(z)$ sowie ihren Betrag

$$|f(z)| = |z^m - 1| = |(x + \imath\, y)^m - 1|$$

als Funktion der reellen Koordinaten x, y.

```
> af:=unapply(abs((x+y*I)^m-1),x,y);
```

(1) 3D-Graphik von $|f(z)|$, $\Re(f(z))$, $\Im(f(z))$ für $m = 5$

```
> pl1:=plot3d(af(x,y),x=-1.2..1.2,y=-1.2..1.2,grid=[41,41],axes=normal,
              orientation=[60,75]):
  pl2:=pointplot3d([seq([cos(k/m*2*Pi),sin(k/m*2*Pi),0],k=0..m-1)],
              color=black,symbol=solidcircle,symbolsize=20):
  pl3:=implicitplot3d(x^2+y^2=1,x=-1..1,y=-1..1,z=-0.1..0,color=blue):
> display(pl1,pl2,pl3,title='|f(z)|=|z^m-1|, m=5',
                      titlefont=[TIMES,BOLD,9]);
```

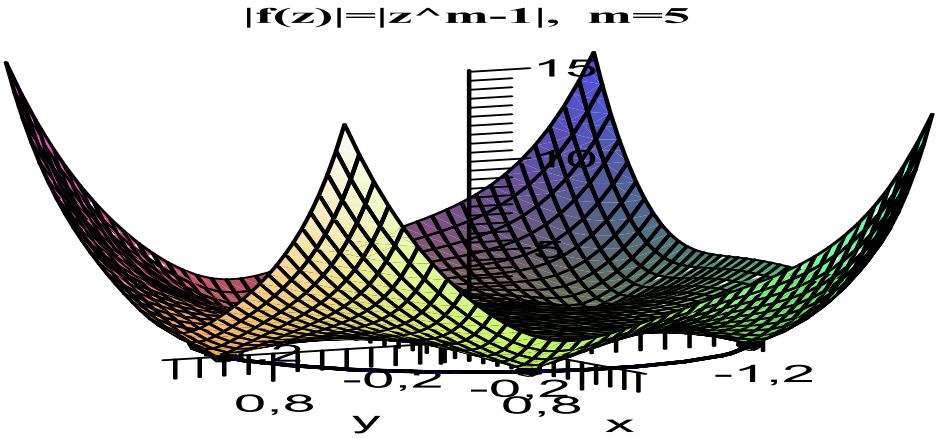
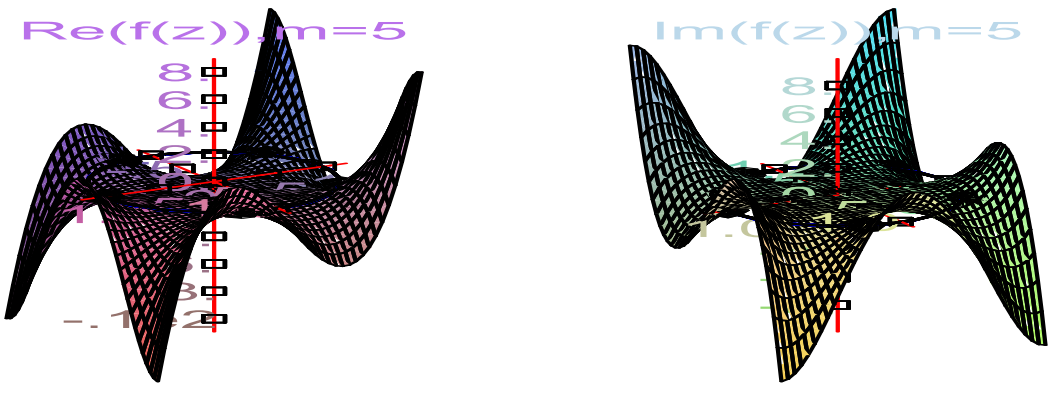


Abb. 3.1 $|f(z)|$ für $m = 5$, dazu Einheitskreis in (x, y)-Ebene, dazu Nullstellen auf dem Rand des Kreises

```
> pp:=array(1..2,[ ]):
> pl1r:=plot3d(Re(f(x+y*I)),x=-1.2..1.2,y=-1.2..1.2,
               grid=[41,41],axes=normal,
               orientation=[60,75],view=[-1.2..1.2,-1.2..1.2,-11..11]):
  display(pl1r,pl2,pl3,title='Re(f(z)), m=5',titlefont=[TIMES,BOLD,8]);
  pp[1]:=display(pl1r,pl2,pl3,title='Re(f(z)), m=5'):
  pl1i:=plot3d(Im(f(x+y*I)),x=-1.2..1.2,y=-1.2..1.2,
               grid=[41,41],axes=normal,
               orientation=[60,75],view=[-1.2..1.2,-1.2..1.2,-11..11]):
  display(pl1i,pl2,pl3,title='Im(f(z)), m=5',titlefont=[TIMES,BOLD,8]);
  pp[2]:=display(pl1i,pl2,pl3,title='Im(f(z)), m=5'):
  display(pp);
```



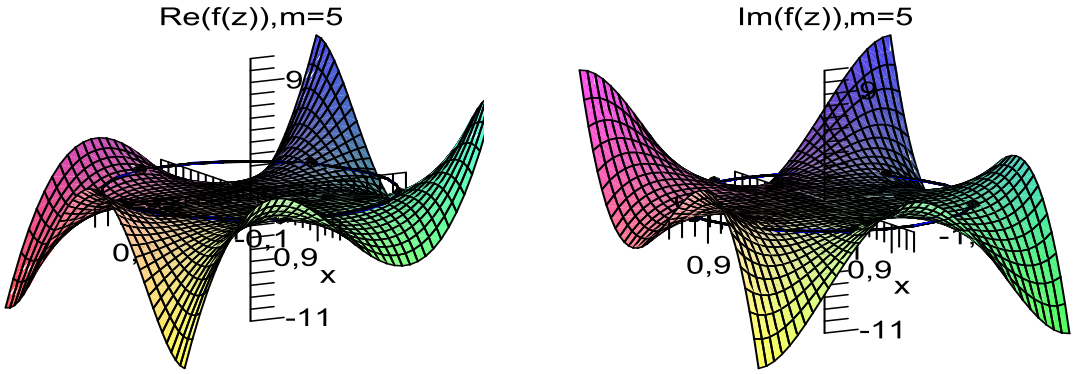


Abb. 3.2 $\Re(f(z))$, $\Im(f(z))$ für $m = 5$,
oben: als Array-Plot nicht so anschaulich,
unten: günstiger als Einzelbilder nebeneinander

(2) 2D-Graphik mit Konturen von $|f(z)|$, $\Re(f(z))$, $\Im(f(z))$ für $m = 5$

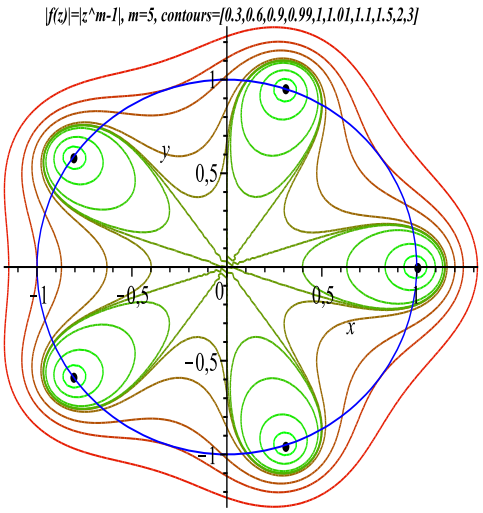


Abb. 3.3
Höhenlinienbild von $|f(z)|$ für $m = 5$,
`contours=[0.3,0.6,0.9,0.99,1,`
`1.01,1.1,1.5,2,3]`

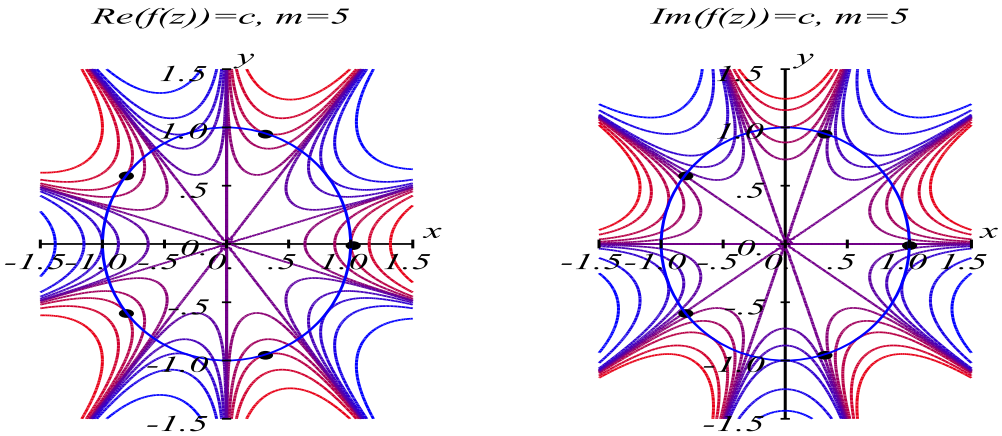


Abb. 3.4 Höhenlinienbilder von $\Re(f(z))$ (links), $\Im(f(z))$ (rechts) für $m = 5$,
$\Re(f(z))$: `contours=[-6,-3,-2,-1.5,-1.1,-1,-0.9,-0.5,0,1,3]`,
$\Im(f(z))$: `contours=[-6,-3,-1,-0.5,-0.2,0,0.2,0.5,1,2,3]`

(3) Iterationsverläufe des Newton-Verfahrens mit Höhenlinienbild von $|f(z)|$
Die Einzugsbereiche der Nullstellen sind angedeutet. Die Grenzen dieser liegen natürlich nicht genau auf den gestrichelten Linien, sondern sind Grenzbereiche und ähneln bei genauerer Betrachtung fraktalen Strukturen. Dort wo der Startvektor irgendwo in einem solchen Bereich liegt, kann die Iterationsfolge durchaus gegen eine entfernte Nullstelle tendieren.

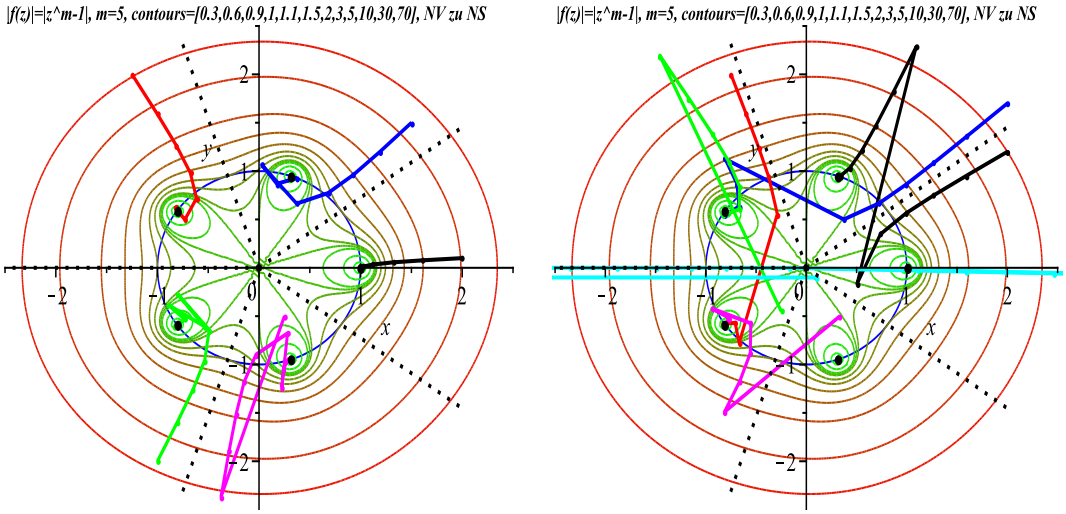


Abb. 3.5 Je 5 Iterationsverläufe des Newton-Verfahrens für $f(z) = 0,\ m = 5$, links: Startpunkte aus der Mitte der Einzugsbereiche der Nullstellen, rechts: Startpunkte aus Grenzbereichen

Iterationsverläufe des Newton-Verfahrens: je 2 Beispiele zu Abb. 3.5

```
> zstart:=2+0.1*I:
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,9,ja):
  ziter1:=[zv]:
```

```
Iterationsverlauf
k =  0   z = +2.0000000000000000e+00 +1.0000000000000000e-01i,  f(z) = +3.020e+01 +7.960e+00i, |f(z)|= +3.123e+01
k =  1   z = +1.6121902180489687e+00 +7.7531032416526038e-02i,  f(z) = +9.640e+00 +2.607e+00i, |f(z)|= +9.986e+00
k =  2   z = +1.3186779990764643e+00 +5.6395379837549907e-02i,  f(z) = +2.915e+00 +8.495e-01i, |f(z)|= +3.036e+00
k =  3   z = +1.1198820295735218e+00 +3.3904634016475416e-02i,  f(z) = +7.453e-01 +2.661e-01i, |f(z)|= +7.914e-01
k =  4   z = +1.0219010300457914e+00 +1.1795270217010213e-02i,  f(z) = +1.129e-01 +6.430e-02i, |f(z)|= +1.299e-01
k =  5   z = +1.0006746212317194e+00 +9.7438403240233952e-04i,  f(z) = +3.368e-03 +4.885e-03i, |f(z)|= +5.934e-03
k =  6   z = +9.9999901782663943e-01 +2.6277305001575543e-06i,  f(z) = -4.911e-06 +1.314e-05i, |f(z)|= +1.403e-05
k =  7   z = +9.9999999998811932e-01 -1.0323505424597729e-11i,  f(z) = -5.940e-11 -5.162e-11i, |f(z)|= +7.870e-11
k =  8   z = +1.0000000000000000e+00 +4.9060121196018292e-22i,  f(z) = +3.458e-22 +2.453e-21i, |f(z)|= +2.477e-21
k =  9   z = +1.0000000000000000e+00 +1.3570375594000000e-43i,  f(z) = +0.000e+00 +6.785e-43i, |f(z)|= +6.785e-43
delta =  4.954508205381844582 65e-22
```

```
> zstart:=1.5+1.5*I;
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,12,ja):
  ziter2:=[zv]:
```

```
Iterationsverlauf
k =  0   z = +1.5000000000000000e+00 +1.5000000000000000e+00i,  f(z) = -3.138e+01 -3.038e+01i, |f(z)|= +4.367e+01
k =  1   z = +1.1901234567901235e+00 +1.2000000000000000e+00i,  f(z) = -1.055e+01 -9.950e+00i, |f(z)|= +1.450e+01
k =  2   z = +9.2758926318323066e-01 +9.6040515033051413e-01i,  f(z) = -3.729e+00 -3.250e+00i, |f(z)|= +4.947e+00
k =  3   z = +6.7929817423833181e-01 +7.7269505816632059e-01i,  f(z) = -1.516e+00 -1.031e+00i, |f(z)|= +1.833e+00
k =  4   z = +3.7079546660139621e-01 +6.6351740872551628e-01i,  f(z) = -8.581e-01 -2.103e-01i, |f(z)|= +8.835e-01
k =  5   z = +2.6557362540656899e-02 +1.0656741046766388e+00i,  f(z) = -8.290e-01 +1.366e+00i, |f(z)|= +1.598e+00
k =  6   z = +1.7535654751804061e-01 +8.6794935122647347e-01i,  f(z) = -5.429e-01 +2.956e-01i, |f(z)|= +6.181e-01
k =  7   z = +3.6753775953382530e-01 +9.2713709197896513e-01i,  f(z) = -6.223e-02 -3.069e-01i, |f(z)|= +3.132e-01
k =  8   z = +3.0637857565906117e-01 +9.4348247921238762e-01i,  f(z) = -3.946e-02 +8.172e-04i, |f(z)|= +3.946e-02
k =  9   z = +3.0906263586643063e-01 +9.5117903845220151e-01i,  f(z) = +6.533e-04 -2.775e-05i, |f(z)|= +6.539e-04
k = 10   z = +3.0901700765444529e-01 +9.5105654779081193e-01i,  f(z) = +1.703e-07 -1.448e-08i, |f(z)|= +1.709e-07
k = 11   z = +3.0901699437494851e-01 +9.5105651629515564e-01i,  f(z) = +1.152e-14 -1.973e-15i, |f(z)|= +1.168e-14
k = 12   z = +3.0901699437494742e-01 +9.5105651629515357e-01i,  f(z) = +5.140e-29 -1.820e-29i, |f(z)|= +5.453e-29
delta =  2.336642958842348741 21e-15
```

```
> zstart:=2+1.2*I;
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,16,ja):
  ziter1:=[zv]:
```

```
Iterationsverlauf
k =  0   z = +2.0000000000000000e+00 +1.2000000000000000e+00i,  f(z) = -6.346e+01 +2.937e+01i, |f(z)|= +6.993e+01
k =  1   z = +1.5962350411872463e+00 +9.5438763903688893e-01i,  f(z) = -2.106e+01 +9.622e+00i, |f(z)|= +2.316e+01
k =  2   z = +1.2677608147771171e+00 +7.4956966257551016e-01i,  f(z) = -7.172e+00 +3.149e+00i, |f(z)|= +7.833e+00
k =  3   z = +9.9144640485704518e-01 +5.6375441539281841e-01i,  f(z) = -2.639e+00 +1.019e+00i, |f(z)|= +2.829e+00
k =  4   z = +7.3676876402603623e-01 +3.4712048564471461e-01i,  f(z) = -1.211e+00 +2.894e-01i, |f(z)|= +1.245e+00
k =  5   z = +5.0340062835770714e-01 -1.6864229796336278e-01i,  f(z) = -1.002e+00 -4.213e-02i, |f(z)|= +1.003e+00
k =  6   z = +1.0930879761095016e+00 +2.2861798946868254e+00i,  f(z) = +8.160e+01 -6.400e+01i, |f(z)|= +1.037e+02
k =  7   z = +8.7344415017058451e-01 +1.8336843371134147e+00i,  f(z) = +2.648e+01 -2.097e+01i, |f(z)|= +3.378e+01
k =  8   z = +6.9633598971885262e-01 +1.4784479258784221e+00i,  f(z) = +8.418e+00 -6.868e+00i, |f(z)|= +1.086e+01
k =  9   z = +5.5177606837067467e-01 +1.2102945791352836e+00i,  f(z) = +2.510e+00 -2.240e+00i, |f(z)|= +3.364e+00
k = 10   z = +4.3249201659930447e-01 +1.0315000035850280e+00i,  f(z) = +6.025e-01 -7.047e-01i, |f(z)|= +9.271e-01
k = 11   z = +3.4378526474744275e-01 +9.5296786595439074e-01i,  f(z) = +5.346e-02 -1.703e-01i, |f(z)|= +1.785e-01
k = 12   z = +3.1012369972867253e-01 +9.4897012655019251e-01i,  f(z) = -8.213e-03 -8.431e-03i, |f(z)|= +1.177e-02
k = 13   z = +3.0900623123119275e-01 +9.5105958600653579e-01i,  f(z) = -2.034e-06 +5.592e-05i, |f(z)|= +5.596e-05
k = 14   z = +3.0901699431502377e-01 +9.5105651605188841e-01i,  f(z) = -1.249e-09 -9.091e-11i, |f(z)|= +1.253e-09
k = 15   z = +3.0901699437494742e-01 +9.5105651629515357e-01i,  f(z) = +6.211e-19 +9.087e-20i, |f(z)|= +6.277e-19
k = 16   z = +3.0901699437494742e-01 +9.5105651629515357e-01i,  f(z) = -1.000e-32 -7.778e-33i, |f(z)|= +1.267e-32
delta =  1.255375633319404892 78e-19
```

```
> zstart:=0.1-0.1*I;
  zv:=zstart:
  zs:=Newton_komplex(f,zstart,61,ja):
  ziter6:=[zv]:
```

```
Iterationsverlauf
k =  0   z = +1.0000000000000000e-01 -1.0000000000000000e-01i,  f(z) = -1.000e+00 +4.000e-05i, |f(z)|= +1.000e+00
k =  1   z = -4.9992000000000000e+02 -8.0000000000000000e-02i,  f(z) = -3.123e+13 -2.498e+10i, |f(z)|= +3.123e+13
k =  2   z = -3.9993599999999680e+02 -6.4000000000002050e-02i,  f(z) = -1.023e+13 -8.187e+09i, |f(z)|= +1.023e+13
k =  3   z = -3.1994879999998962e+02 -5.1200000000006644e-02i,  f(z) = -3.353e+12 -2.683e+09i, |f(z)|= +3.353e+12
k =  4   z = -2.5595903999997261e+02 -4.0960000000017532e-02i,  f(z) = -1.099e+12 -8.790e+08i, |f(z)|= +1.099e+12
k =  5   z = -2.0476723199993149e+02 -3.2768000000043852e-02i,  f(z) = -3.600e+11 -2.880e+08i, |f(z)|= +3.600e+11
k = 10   z = -6.7098126575748909e+01 -1.0737418243847714e-02i,  f(z) = -1.360e+09 -1.088e+06i, |f(z)|= +1.360e+09
k = 20   z = -7.2045612576152531e+00 -1.1529504532572208e-03i,  f(z) = -1.941e+04 -1.553e+01i, |f(z)|= +1.941e+04
k = 30   z = -1.6542726025588496e-01 -8.9242182235452586e-04i,  f(z) = -1.000e+00 -3.342e-06i, |f(z)|= +1.000e+00
k = 40   z = +3.5815402921261657e+01 -7.7343759261263931e-01i,  f(z) = +5.866e+07 -6.357e+06i, |f(z)|= +5.900e+07
k = 50   z = +3.8462040287330430e+00 -8.2999234485342323e-02i,  f(z) = +8.368e+02 -9.073e+01i, |f(z)|= +8.417e+02
...
k = 60   z = +9.9999999999938286e-01 -4.4550916999697454e-12i,  f(z) = -3.086e-12 -2.228e-11i, |f(z)|= +2.249e-11
k = 61   z = +1.0000000000000000e+00 +1.0997615464768323e-23i,  f(z) = -1.947e-22 +5.499e-23i, |f(z)|= +2.023e-22
delta =  4.497632785075353877 78e-12
```

Noch einige Berechnungen zu den Nullstellen/Wurzeln

```
> Digits:=32:
  ns:=array(1..m,1..2,[]):
  de:=2*Pi/m:
  for k from 1 to m do
   ns[k,1]:=cos(de*(k-1)):
   ns[k,2]:=sin(de*(k-1)):
  end do;
  fprintf(default,`\nm-te Wurzeln aus 1, m = %2.0f\n`,m):
  fprintf(default,`Tabelle der m Wurzeln\n`):
  for k from 1 to m do
   fprintf(default,`ns(%2.0f) =  %+.20e %+.20ei\n`,k,ns[k,1],ns[k,2]):
  end do:
  Digits:=16:
```

$$ns_{1,1} := 1$$
$$ns_{1,2} := 0$$
$$ns_{2,1} := \cos\left(\frac{2}{5}\pi\right)$$
$$ns_{2,2} := \sin\left(\frac{2}{5}\pi\right)$$
$$ns_{3,1} := -\cos\left(\frac{1}{5}\pi\right)$$
$$ns_{3,2} := \sin\left(\frac{1}{5}\pi\right)$$
$$ns_{4,1} := -\cos\left(\frac{1}{5}\pi\right)$$
$$ns_{4,2} := -\sin\left(\frac{1}{5}\pi\right)$$
$$ns_{5,1} := \cos\left(\frac{2}{5}\pi\right)$$
$$ns_{5,2} := -\sin\left(\frac{2}{5}\pi\right)$$

```
m-te Wurzeln aus 1, m =  5
Tabelle der m Wurzeln
ns( 1) =  +1.00000000000000000000e+00 +0.00000000000000000000e+00i
ns( 2) =  +3.09016994374947424102e-01 +9.51056516295153572116e-01i
ns( 3) =  -8.09016994374947424102e-01 +5.87785252292473129169e-01i
ns( 4) =  -8.09016994374947424102e-01 -5.87785252292473129169e-01i
ns( 5) =  +3.09016994374947424102e-01 -9.51056516295153572116e-01i
```

```
> h1:=convert(ns[2,1],RootOf);
  evalf(h1);
  solve(-1+2*Z+4*Z^2,Z);
  h2:=convert(ns[2,2],RootOf);
  evalf(h2);
  solve(5-20*Z^2+16*Z^4,Z);
```

$$h1 := RootOf(-1 + 2\_Z + 4\_Z^2,\ 0.3090169943749474)$$
$$0.3090169943749474$$
$$-\frac{1}{4} + \frac{1}{4}\sqrt{5},\ -\frac{1}{4} - \frac{1}{4}\sqrt{5}$$
$$h2 := RootOf(16\_Z^4 - 20\_Z^2 + 5,\ 0.9510565162951536)$$
$$0.9510565162951536$$
$$-\frac{1}{4}\sqrt{10 + 2\sqrt{5}},\ \frac{1}{4}\sqrt{10 + 2\sqrt{5}},\ -\frac{1}{4}\sqrt{10 - 2\sqrt{5}},\ \frac{1}{4}\sqrt{10 - 2\sqrt{5}}$$

Nun einige Berechnungen zu den Einzugsgebieten der Nullstellen von $f(z) = z^m - 1$, $z = x + \imath y = (x, y)$ in MATLAB.
Dabei wird auf Vorgehensweisen zurückgegriffen, wie sie bei Darstellung von Mandelbrot- oder Julia-Mengen - im Allgemeinen bei Fraktalen - zu finden sind.
Man wählt ein Gebiet von Startwerten, meist ein Rechteckgebiet, das in der Regel auch den Einheitskreis mit den Nullstellen enthält. Dazu werden die Iterationsfolgen mit dem Newton-Verfahren

$$\begin{aligned} z_{k+1} &= z_k - \frac{f(z_k)}{f'(z_k)} = \frac{1}{m}\Big((m-1)z_k + \frac{1}{z_k^{m-1}}\Big), \quad k = 0, 1, 2, ..., k_{max}, \\ & z_0 \in \mathrm{G} = [a_u, a_o] \times [b_u, b_o], \end{aligned} \tag{3.1}$$

erzeugt. Dabei werden für jeden Startwert die Aspekte der Konvergenz, der Divergenz (bei Erreichung der maximalen Iterationanzahl bzw. Überschreitung einer Schranke), der Beschränkheit der Folge, der Anzahl der Iterationen bei Konvergenz usw. untersucht. Dies führt für jeden Startwert auf die Festlegung einer Farbe (aus einem vorgegebenen Farbspektrum) in Abhängigkeit von der Iterationszahl und somit insgesamt auf ein eingefärbtes Gebiet. Dort wo das Newton-Verfahren einmal konvergiert, tut es dies meist sehr schnell. Konvergenz bedeutet, dass der Startwert zum Einzugsbereich einer Nullstelle gehört. Dazu kommt noch die Besonderheit, dass alle Nullstellen verschieden und betragsmäßig gleich Eins sind, sowie der Umstand, dass die Ränder der Einzugsbereiche der Nullstellen an fraktale Strukturen erinnern.
Die einzige kritische Stelle, wo das Newton-Verfahren nicht rechnen kann, ist der Koordinatenursprung. Startet man nahe $(0, 0)^T$, so können die Iterierten betragsmäßig extrem groß werden, und die Iterationsfolge konvergiert erst nach sehr vielen Schritten zu einer Nullstelle. Bei moderater Größe der maximalen Iterationsanzahl ist die dann zu klein, um die Konvergenz zu erkennen und der Startpunkt wird als Stelle mit Divergenzcharakter eingestuft. Ähnlich, aber nicht ganz so extrem gestaltet sich die Situation in den fraktalen Grenzbereichen.
Somit treten in der Modellierung und Simulation von Berechnungen zahlreiche Parameter auf, wie die Festlegung von verschiedenen Schranken. Dem Leser öffnet sich damit ein weites Betätigungsfeld. Hier sollen nur wenige Situationen und Fälle verwendet und beschrieben werden.

Das MATLAB-Skriptfile enthält 4 Varianten, die in jedem Fall einführend erläutert werden. Es wurde für $m = 5$ gerechnet und damit sind auch die Abbildungen erzeugt worden.

```
% einzugsbereich_nv1.m
% Konvergenzbereich des NV fuer f(z)=z^m-1=0, m-te Wurzeln aus 1, m>=1
%
% Standardalgorithmus im Komplexen
% Newton-Verfahren als Fixpunktformel fuer f(z)=0, z=x+i*y
%   z(k+1)=z(k)-f(z(k))/f'(z(k))
%   z(k+1)=1/m[(m-1)z(k)+1/z(k)^(m-1)]=g(z(k))
```

```
% Auswahl eines Wertes m
% Startpunkte  z aus [au,ao]x[bu,bo] (Matrix)
% Abbruch, falls x^2+y^2>=Schranke oder max. Iterationszahl erreicht ist
% aus Iterationsanzahl Berechnung der Farbe des Punktes
%
% Falls fuer alle Iterationen gilt x^2+y^2<Schranke, dann liegt der Punkt z
% im Einzugsbereich (Innenbereich)
% Achtung  |Wurzel|=1
%
% Mehrfarbig:  Einzugsbereich  bordeaux, rote Farbe
%              sonst           blau,...

diary ebereich1.txt       % Protokolldatei
diary off
echo off
clear all
clc
clf
format compact
format short

fprintf(1,'\nKonvergenzbereich des Newton-Verfahrens    ');
fprintf(1,'\nfuer f(z)=z^m-1=0, m-te Wurzeln aus 1, m>=1');
fprintf(1,'\n4 Varianten fuer Beispiel m = ...');
fprintf(1,'\n=========================================\n\n');
x0 = -4; y0 = -4; dx = 8; dy = 8;
n = 256;           % 128
maxit1 = 128;      % 256
schranke1 = 16;    % 16,10,4,2,1.1
maxit2 = 20;
schranke2 = 1.1;
maxit3 = 64;
schranke3 = 2;

% 4 Varianten fuer Beispiel m = ...
x = linspace(x0,x0+dx,n); y = linspace(y0,y0+dy,n);
[x,y] = meshgrid(x,y);
m = 5;
m = input('m = ');
w = 0:2*pi/m:2*pi;
pc = cos(w); ps = sin(w);
ns = []; for k=1:m, ns(k,1:2)=[pc(k),ps(k)]; end
disp(' ');
disp('m Wurzeln (Nullstellen)');
ns
ww = 0:0.01:2*pi;
```

```
% Variante 1
% Startwerte z mit |z|^2>=schranke1 werden von vornherein als nicht zum
% Einzugsbereich gehoerend eingestuft.
% Nach wenigen Anfangsiterationen zeigen sich diejenigen Startwerte, die
% sogenannte Ausreisser (Pseudo-Divergenz der Iteration wegen ev. zu kleiner
% maximaler Iterationsanzahl oder zu kleiner Schranke) sind und die find-
% Bedingung nicht erfuellen. Solche Werte liegen in und nahe z=0 und an den
% Grenzen der Einzugsbereiche der m Nullstellen speichenfoermig angeordnet.
% Damit erkennt man im Groben auch diese Einzugsbereiche.
% Die anderen Startwerte z mit |z|^2<schranke1 werden im Iterationsverlauf
% die find-Bedingung erfuellen (bordeaux Farbe) und der Betrag der Iterierten
% konvergiert gegen 1 (Betrag der Nullstellen).

z = x+y*1i;               % Startfeld
K = ones(n,n);            % Farbfeld mit Werten aus 1..maxit
for k = 1:maxit1
    a = find((real(z).^2+imag(z).^2)<schranke1);
            % Vektor mit Indizes bei erfuellter Bedingung
    z(a) = ((m-1)*z(a)+z(a).^(1-m))/m;
    K(a) = k;
end
figure(1);
clf
colormap(jet(maxit1)) ;
image(x0+[0 dx],y0+[0 dy],K);
set(gca,'YDir','normal');
axis equal
axis tight
hold on
plot(pc,ps,'o:','LineWidth',1,'Color','g', ...
     'MarkerSize',6,'MarkerEdgeColor','k','MarkerFaceColor','g')
plot(cos(ww),sin(ww),'y-')
plot([x0,x0+dx],[0,0],'k-',[0,0],[y0,y0+dy],'k-')
hold off
print -dpsc 'ebereich101.ps'
print -djpeg 'ebereich101.jpg'

% Variante 2
% Fuer alle Startwerte z ausser Null werden alle Iterationen durchgefuehrt.
% Man benoetigt beim Newton-Verfahren nicht allzu viele Iterationen, um die
% Konvergenz festzustellen. Deshalb braucht man fuer die letzten Iterierten
% nur den Test darauf zu machen, ob ihr Betrag nahe der 1 liegt. Ist das der
% Fall, gehoeren die Werte z zum Einzugsbereich der m Nullstellen (bordeaux
% Farbe). Die Werte, welche die find-Bedingung nicht erfuellen, sind mit der
% Pseudo-Divergenz der Iteration verbunden (siehe Variante 1). Solche Werte
% liegen in und nahe z=0 und an den Grenzen der Einzugsbereiche
```

```
% der m Nullstellen speichenfoermig angeordnet.
% Damit erkennt man im Groben auch diese Einzugsbereiche.

z = x+y*1i;                % Startfeld
K = ones(n,n);             % Farbfeld mit Werten aus 1..maxit
for k = 1:maxit2
    z = ((m-1)*z+z.^(1-m))/m;
end
a = find((real(z).^2+imag(z).^2)<schranke2);
        % Vektor mit Indizes bei erfuellter Bedingung
K(a) = maxit2;
figure(2);
clf
colormap(jet(maxit2));
image(x0+[0 dx],y0+[0 dy],K);
set(gca,'YDir','normal');
axis equal
axis tight
hold on
plot(pc,ps,'o:','LineWidth',1,'Color','g', ...
     'MarkerSize',6,'MarkerEdgeColor','k','MarkerFaceColor','g')
plot(cos(ww),sin(ww),'y-')
plot([x0,x0+dx],[0,0],'k-',[0,0],[y0,y0+dy],'k-')
hold off
print -dpsc 'ebereich102.ps'
print -djpeg 'ebereich102.jpg'
```

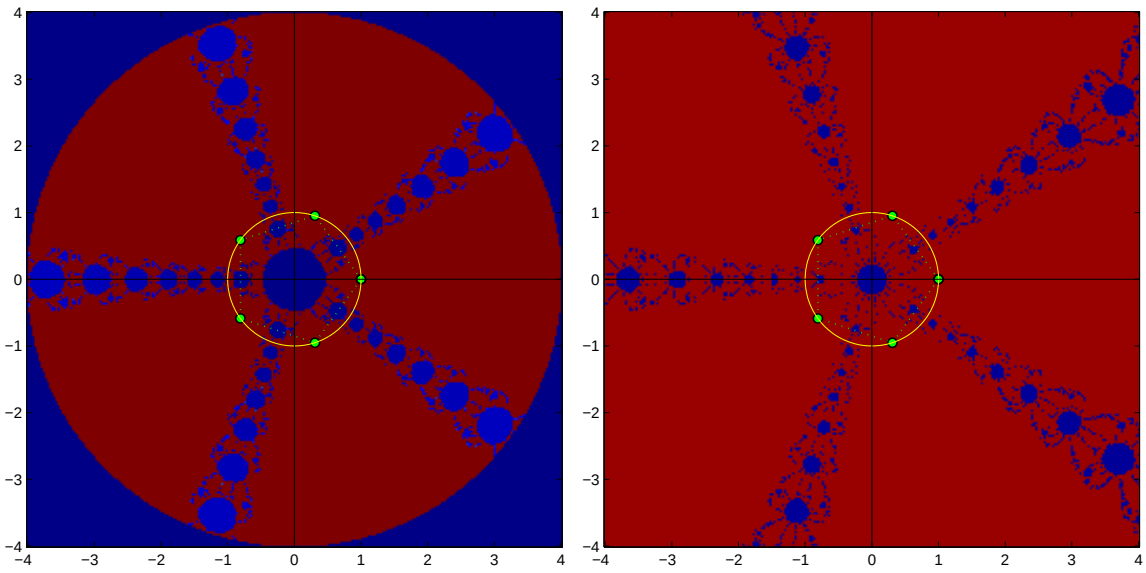


Abb. 3.6 Nullstellen, Einzugsbereiche und Grenzen, $m = 5$, Variante 1 links, Variante 2 rechts

```
% Variante 3
% Fuer alle Startwerte z ausser Null werden alle Iterationen durchgefuehrt.
% Man benoetigt beim Newton-Verfahren nicht allzu viele Iterationen, um
% Konvergenz festzustellen. Nach wenigen Anfangsiterationen zeigen sich
% diejenigen Startwerte, die sogenannte Ausreisser (Pseudo-Divergenz der
% Iteration) sind und die find-Bedingung dann stets erfuellen. Solche Werte
% liegen in und nahe z=0 und an den Grenzen der Einzugsbereiche der
% m Nullstellen speichenfoermig angeordnet. Sie haben eine ganz andere Farbe.
% Bei anderen Startwerten braucht die Iteration ev. etwas laenger, damit
% |z|^2>schranke3 wird. Somit werden grob die Grenzen der Einzugsbereiche
% deutlich.
% Im Einzugsbereich liegen Startwerte z nahe den Nullstellen und welche
% damit die find-Bedingung nicht erfuellen (bordeaux Farbe).
% Weitere Startwerte kommen schnell zum Einzugsbereich hinzu (rote Farbe).

z = x+y*1i;               % Startfeld
K = maxit3*ones(n,n) ;    % Farbfeld mit Werten aus 1..maxit
for k = 1:maxit3
    a = find((real(z).^2+imag(z).^2)>schranke3);
          % Vektor mit Indizes bei noch nicht erfuellter Bedingung
    z = ((m-1)*z+z.^(1-m))/m;
    K(a) = maxit3-k+1;
end
figure(3);
clf
colormap(jet(maxit3));
image(x0+[0 dx],y0+[0 dy],K);
set(gca,'YDir','normal');
axis equal, axis tight
hold on
plot(pc,ps,'o:','LineWidth',1,'Color','g', ...
     'MarkerSize',6,'MarkerEdgeColor','k','MarkerFaceColor','g')
plot(cos(ww),sin(ww),'y-')
plot([x0,x0+dx],[0,0],'k-',[0,0],[y0,y0+dy],'k-')
hold off
print -dpsc 'ebereich103.ps'
print -djpeg 'ebereich103.jpg'

% Variante 4
% wie Variante 3, aber Einzugsgebiete fuer jede Wurzel anders faerben

z = x+y*1i;               % Startfeld
K = maxit3*ones(n,n);     % Farbfeld mit Werten aus 1..maxit
for k = 1:maxit3
    a = find((real(z).^2+imag(z).^2)>schranke3);
          % Vektor mit Indizes bei noch nicht erfuellter Bedingung
```

```
    z = ((m-1)*z+z.^(1-m))/m;
    K(a) = maxit3-k+1;
end
figure(4);
clf
colormap(jet(maxit3));
image(x0+[0 dx],y0+[0 dy],K);
set(gca,'YDir','normal');
axis equal, axis tight
hold on
KK = ones(n,n);
colormap(jet(m+1));     % fuer Farbunterschiede
for k=1:m
  hrz = real(z)-ns(k,1); hiz = imag(z)-ns(k,2);
  ah = find((hrz.^2+hiz.^2)<0.001);
  KK(ah) = k+1;
  image(x0+[0 dx],y0+[0 dy],KK);
end
plot(pc,ps,'ko:','LineWidth',1,'Color','k', ...
     'MarkerSize',6,'MarkerEdgeColor','k','MarkerFaceColor','g')
plot(cos(ww),sin(ww),'k-')
plot([x0,x0+dx],[0,0],'k-',[0,0],[y0,y0+dy],'k-')
hold off
print -dpsc 'ebereich104.ps'
print -djpeg 'ebereich104.jpg'
pause, format, diary off, echo off
```

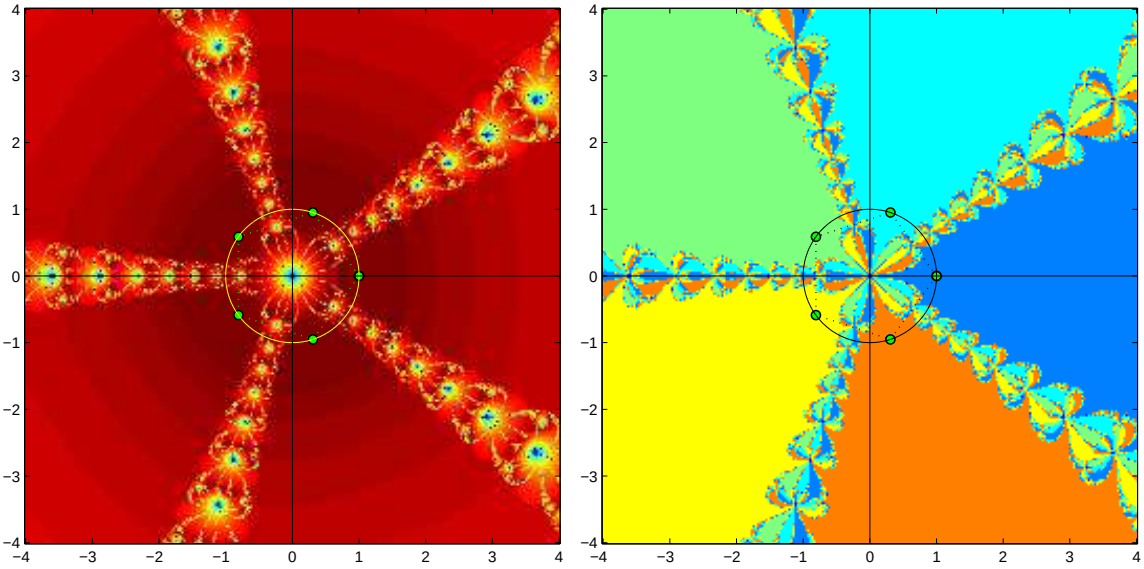


Abb. 3.7 Nullstellen, Einzugsbereiche und Grenzen, $m = 5$, Variante 3 links, Variante 4 rechts

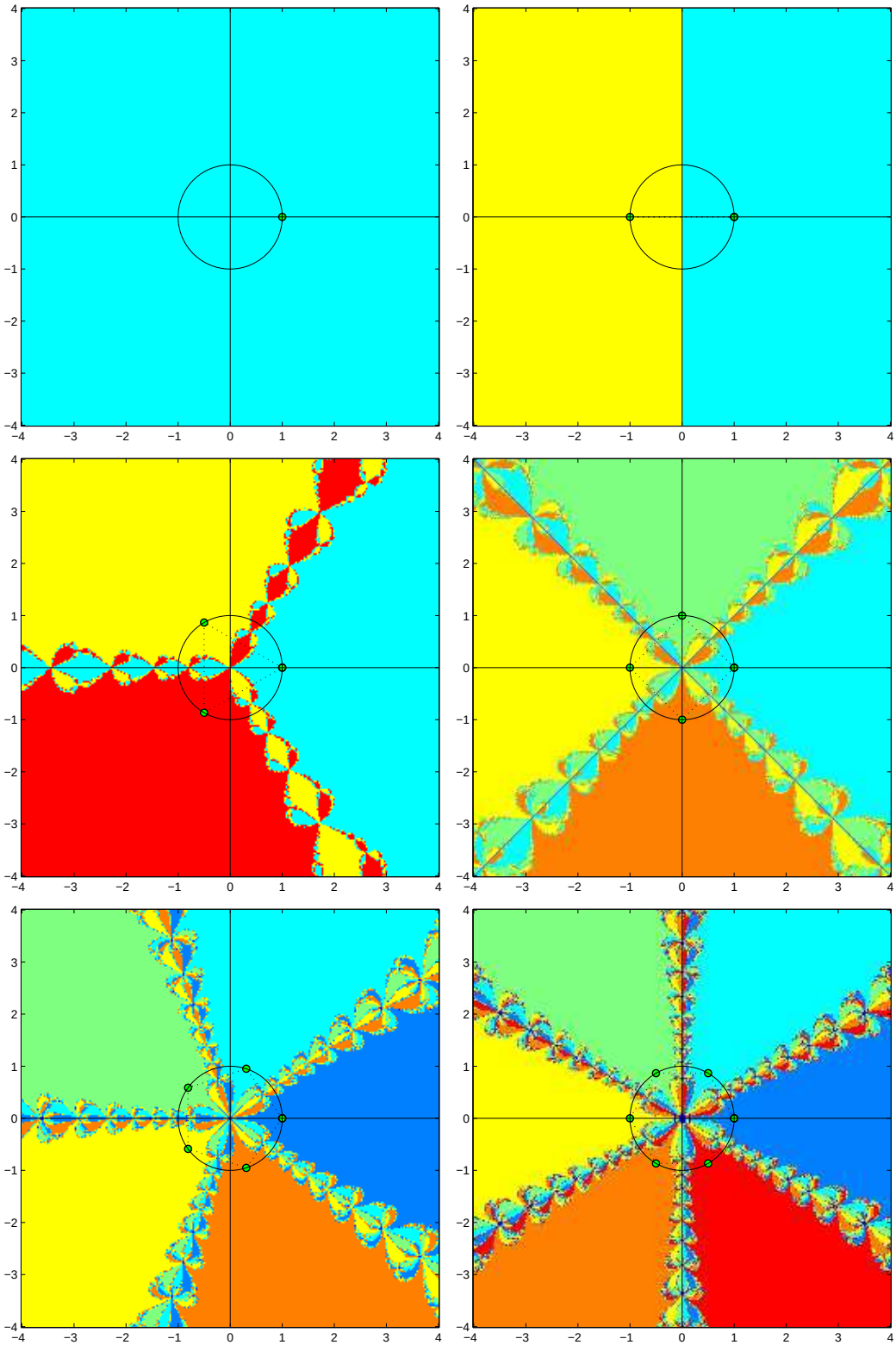


Abb. 3.8 Nullstellen, Einzugsbereiche und Grenzen, $m = 1(1)6$, l.o.n.r.u.

3.2 Wurzeln und Koordinatensysteme

Betrachten wir die Nullstellenaufgabe in Polarkoordinaten (1.16), (1.17) für

$$\begin{aligned} f(z) &= z^m - 1 \;=\; (x + \imath\, y)^m - 1 \\ &= \sum_{k=0}^{m} \binom{m}{k} x^{m-k}\, (\imath\, y)^k - 1 \\ &= \sum_{k=0(2)}^{m} \binom{m}{k} (-1)^{k/2} x^{m-k}\, y^k - 1 + \imath \sum_{k=1(2)}^{m} \binom{m}{k} (-1)^{(k-1)/2} x^{m-k}\, y^k \\ &= \Re(f(x + \imath\, y)) + \imath\, \Im(f(x + \imath\, y)) \\ &= (r e^{\imath\, \varphi})^m - 1 \\ &= r^m \cos(m\varphi) - 1 + \imath\, r^m \sin(m\varphi) \end{aligned} \tag{3.2}$$

Real- und Imaginärteil von $f(z)$ sind zugleich die Koordinatenfunktionen $u(r, \varphi)$ und $v(r, \varphi)$, so dass sich das nichtlineare Gleichungssystem

$$\begin{aligned} u(r, \varphi) &= r^m \cos(m\varphi) - 1 \;=\; 0, \quad 0 \le r \le R,\ 0 \le \varphi \le 2\pi, \\ v(r, \varphi) &= r^m \sin(m\varphi) \qquad\quad =\; 0. \end{aligned} \tag{3.3}$$

ergibt.
Wir notieren noch für $m = 1(1)6$ die Funktion $f(z)$ explizit mit ihrem Real- und Imaginärteil als Polynome von x, y. Insbesondere werden in Beispielrechnungen die Gleichungen $z^m - 1 = 0,\ m = 5, 6$, einbezogen.

$$\begin{aligned} z - 1 &= x - 1 + \imath\, y, \\ z^2 - 1 &= x^2 - y^2 - 1 + \imath\, 2xy, \\ z^3 - 1 &= x^3 - 3xy^2 - 1 + \imath\, (3x^2 y - y^3), \\ z^4 - 1 &= x^4 - 6x^2 y^2 + y^4 - 1 + \imath\, (4x^3 y - 4xy^3), \\ z^5 - 1 &= x^5 - 10x^3 y^2 + 5xy^4 - 1 + \imath\, (5x^4 y - 10x^2 y^3 + y^5), \\ z^6 - 1 &= x^6 - 15x^4 y^2 + 15x^2 y^4 - y^6 - 1 + \imath\, (6x^5 y - 20x^3 y^3 + 6xy^5). \end{aligned}$$

Parallel zum System (3.3) werden wir auch das Gleichungssystem

$$F(r, \varphi) = (F_1(r, \varphi), F_2(r, \varphi))^T = (u(r, \varphi)^2,\ v(r, \varphi)^2)^T = (0, 0)^T \tag{3.4}$$

in Verbindung mit dem Newton-Verfahren bzw. das zu minimierende Funktional

$$h(r, \varphi) = u(r, \varphi)^2 + v(r, \varphi)^2 \ge 0, \quad h(r, \varphi) \to \min_{r, \varphi}, \tag{3.5}$$

im Zusammenhang mit dem modifizierten Gradientenverfahren [9], [10] betrachten.

Zur Lösung des nichtlinearen Gleichungssystems (3.3) bei gegebenem m bezüglich der Variablen r und φ, die erst einmal beliebig sein können, sind folgende Aspekte zu beachten.

- Wir haben die spezielle Lösung und Einheitswurzel $(r, \varphi) = (1, 0)$, die auch als Sonderfall in anderen Varianten in Erscheinung tritt.
- Es können wegen $r^m \cos(m\varphi) = 1$ nur Lösungen mit $|r| \geq 1$ auftreten.
- Die Lösungsmannigfaltigkeit kann i. Allg. komplexe Lösungen enthalten.
- Die Lösungsmenge wird mehr als nur die m-ten Einheitswurzeln umfassen.
- Für die Beschreibung der Lösungsmannigfaltigkeit braucht man Fallunterscheidungen bezüglich der Größe m als gerade bzw. ungerade Zahl sowie des Vorzeichens der Variablen r.
- Anwendung der Maple-Kommandos `solve` und `fsolve`.

(a) Untersuchung von komplexen Lösungen

Lösungen (r, φ) mit komplexen Werten r sind zwar nicht relevant, aber sie existieren. Für ganzzahlige Werte k ($k = 0$ ist der Sonderfall $(1, 0)$) notieren wir diese als $(r, \varphi) = (e^{\imath\, k\pi/m},\, k\pi/m)$ mit $|r| = 1$.
Es gelten nämlich

$$\begin{aligned} r^m \cos(m\varphi) - 1 &= \left(e^{\imath\, k\pi/m}\right)^m \cos(mk\pi/m) - 1 \\ &= e^{\imath\, k\pi} \cos(k\pi) - 1 \\ &= [\cos(k\pi) + \imath \sin(k\pi)] \cos(k\pi) - 1 \\ &= \cos^2(k\pi) - 1 \\ &= 0, \end{aligned}$$

und

$$r^m \sin(m\varphi) = r^m \sin(mk\pi/m) = r^m \sin(k\pi) = 0.$$

(b) Als reelle Lösungen kommen nur solche in Frage, wo zunächst $\sin(m\varphi) = 0$ ist. Unter Berücksichtigung der Periode folgen daraus die Winkelwerte

$$\varphi_k = k\,\frac{2\pi}{2m} = k\,\frac{\pi}{m}, \quad k = 0, 1, 2, ..., 2m - 1.$$

Damit erhält man

$$\begin{aligned} r^m \cos(m\varphi_k) &= r^m \cos(mk\pi/m) \;=\; r^m \cos(k\pi) \\ &= r^m\,(-1)^{[(k+1)/2]} \\ &= r^m \cdot \begin{cases} 1 & \text{für } k \text{ gerade,} \\ -1 & \text{für } k \text{ ungerade,} \end{cases} \end{aligned}$$

und weiter mit $0 = r^m \cos(m\varphi_k) - 1,\ k = 0, 1, 2, ..., 2m-1$, die Beziehung

$$1 = r^m \cdot \begin{cases} 1 & \text{für } k \text{ gerade,} \\ -1 & \text{für } k \text{ ungerade.} \end{cases} \tag{3.6}$$

Damit ergibt sich zunächst die Bedingung $|r| = 1$.

(c) Für die reellen Lösungen (r, φ_k) mit $r = \pm 1$ und $\varphi_k = k\pi/m$ machen wir nun Fallunterscheidungen bez. m und r.

$\underline{m \text{ gerade}}$
Die Beziehung (3.6) hat die Gestalt

$$1 = \begin{cases} 1 & \text{für } k \text{ gerade,} \\ -1 & \text{für } k \text{ ungerade,} \end{cases}$$

so dass nur gerade Werte k in Frage kommen.
Lösungen sind damit

$$r = \pm 1,\ \varphi_k = k\pi/m,\ k = 0, 2, 4, ..., 2m-2. \tag{3.7}$$

Dazu gehören die bekannten Einheitswurzeln $r = 1,\ \varphi_k = k\pi/m,\ k = 0, 2, ..., 2m-2$, sowie die anderen Lösungen $r = -1,\ \varphi_k = k\pi/m,\ k = 0, 2, 4, ..., 2m-2$.

$\underline{m \text{ ungerade}}$
Die Beziehung (3.6) hat für $r = 1$ die Gestalt

$$1 = \begin{cases} 1 & \text{für } k \text{ gerade,} \\ -1 & \text{für } k \text{ ungerade,} \end{cases}$$

so dass nur gerade Werte k in Frage kommen.
Lösungen sind wiederum die bekannten Einheitswurzeln

$$r = 1,\ \varphi_k = k\pi/m,\ k = 0, 2, 4, ..., 2m-2. \tag{3.8}$$

Für $r = -1$ hat die Beziehung (3.6) die Form

$$1 = (-1) \cdot \begin{cases} 1 & \text{für } k \text{ gerade} \\ -1 & \text{für } k \text{ ungerade} \end{cases} = \begin{cases} -1 & \text{für } k \text{ gerade,} \\ 1 & \text{für } k \text{ ungerade,} \end{cases}$$

so dass nur ungerade Werte k in Frage kommen. Die anderen Lösungen sind also $r = -1,\ \varphi_k = k\pi/m,\ k = 1, 3, 5, ..., 2m-1$.

Rechnungen und Graphik in Maple

Kommandos `solve` und `fsolve`

```
> m:='m':
  u:=unapply(r^m*cos(m*phi)-1,r,phi):
  v:=unapply(r^m*sin(m*phi),r,phi):

> s:=solve({u(r,phi),v(r,phi)},{r,phi});
  # analog  s:=solve({u(r,phi),v(r,phi)},{r,phi}) assuming r::positive;
```

$$s := \{\varphi = 0,\ r = 1\}, \left\{\varphi = \frac{\pi}{m},\ r = e^{\frac{\mathrm{I}\pi}{m}}\right\}$$

```
> m:=6:
  fsolve({u(r,phi),v(r,phi)},{r,phi});                        # phi=2*Pi
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi=-1..1});         # phi=0
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi= 1..2});         # phi=Pi/3
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi= 2..3});         # phi=2*Pi/3
  ...
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi= 6..7});         # phi=2*Pi

  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=-1..1});       # phi=0
  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=1..2});        # phi=Pi/3
  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=2..3});        # phi=2*Pi/3
  ...
  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=5..6});        # phi=5*Pi/3
```

$$\{\varphi = 6.283185307,\ r = -1.000000000\}$$
$$\{\varphi = 0.,\ r = 1.000000000\}$$
$$\{\varphi = 1.047197551,\ r = 1.000000000\}$$
$$\{\varphi = 2.094395102,\ r = 1.000000000\}$$
$$\ldots$$
$$\{\varphi = 6.283185307,\ r = 1.000000000\}$$

$$\{\varphi = 0.,\ r = -1.000000000\}$$
$$\{\varphi = 1.047197551,\ r = -1.000000000\}$$
$$\{\varphi = 2.094395102,\ r = -1.000000000\}$$
$$\ldots$$
$$\{\varphi = 5.235987756,\ r = -1.000000000\}$$

```
> m:=5:
  fsolve({u(r,phi),v(r,phi)},{r,phi});                        # phi=11*Pi/5
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi=-1..1});         # phi=0
  fsolve({u(r,phi),v(r,phi)},{r=0.8..1.2,phi= 1..2});         # phi=2*Pi/5
  ...

  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=0..1});        # phi=Pi/5
  fsolve({u(r,phi),v(r,phi)},{r=-1.2..-0.8,phi=1..2});        # phi=3*Pi/5
  ...
```

$$\{\varphi = 6.911503838,\ r = -1.000000000\}$$
$$\{\varphi = 0.,\ r = 1.000000000\}$$
$$\{\varphi = 1.256637061,\ r = 1.000000000\}$$
$$\{\varphi = 0.6283185307,\ r = -1.000000000\}$$
$$\{\varphi = 1.884955592,\ r = -1.000000000\}$$

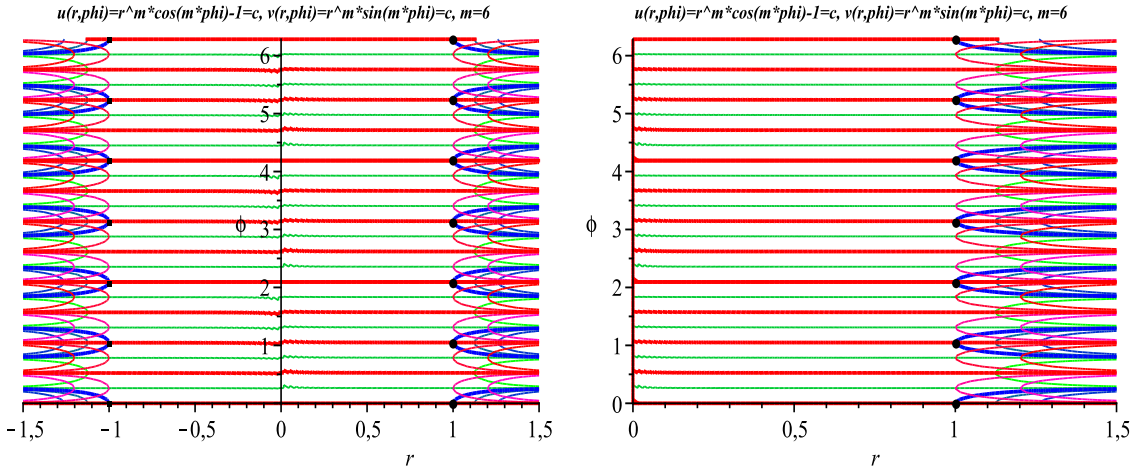


Abb. 3.9 Lösungen von $(u(r,\varphi), v(r,\varphi)) = (0,0)$ mit $r = \pm 1$, $m = 6$, Konturen $u(r,\varphi) = c$, $v(r,\varphi) = c$ mit `contours=[-3,-1,0,1,3]`

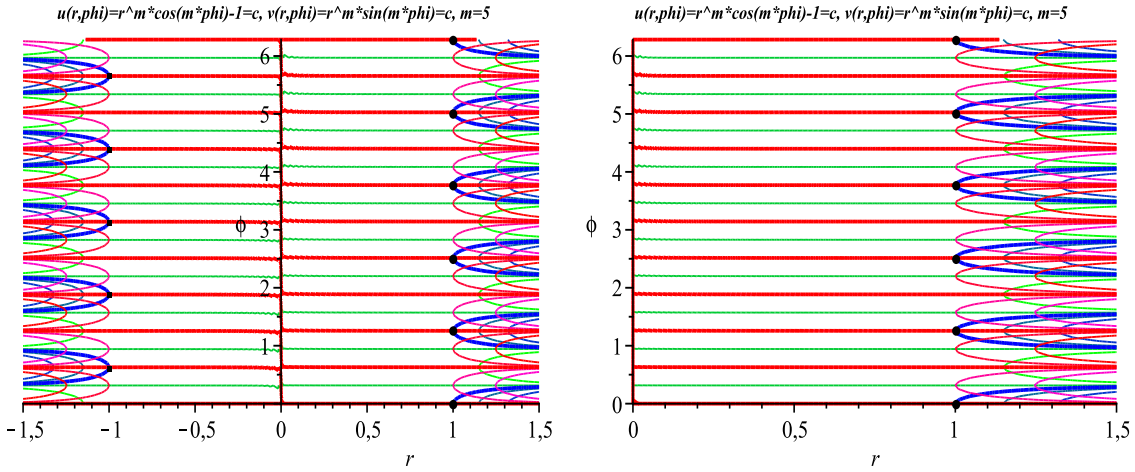


Abb. 3.10 Lösungen von $(u(r,\varphi), v(r,\varphi)) = (0,0)$ mit $r = \pm 1$, $m = 5$, Konturen $u(r,\varphi) = c$, $v(r,\varphi) = c$ mit `contours=[-3,-1,0,1,3]`

Die uns interessierenden Einheitswurzeln liegen auf dem Rand des Einheitskreies und damit auf der Geraden $r = 1$.

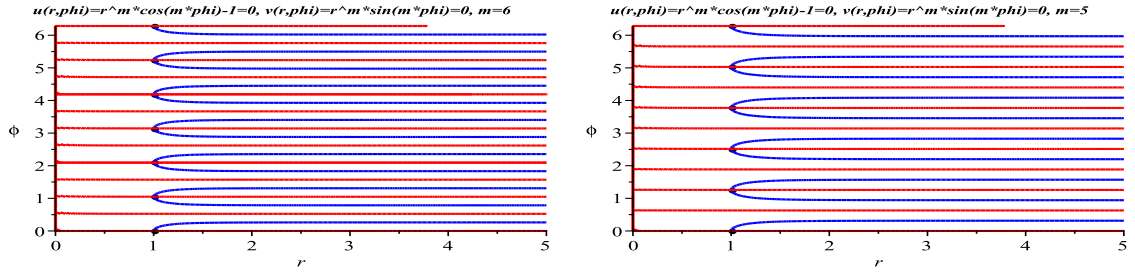


Abb. 3.11 Lösungen von $(u(r,\varphi), v(r,\varphi)) = (0,0)$ mit $r = 1$, $m = 6, 5$, Konturen $u(r,\varphi) = 0$, $v(r,\varphi) = 0$

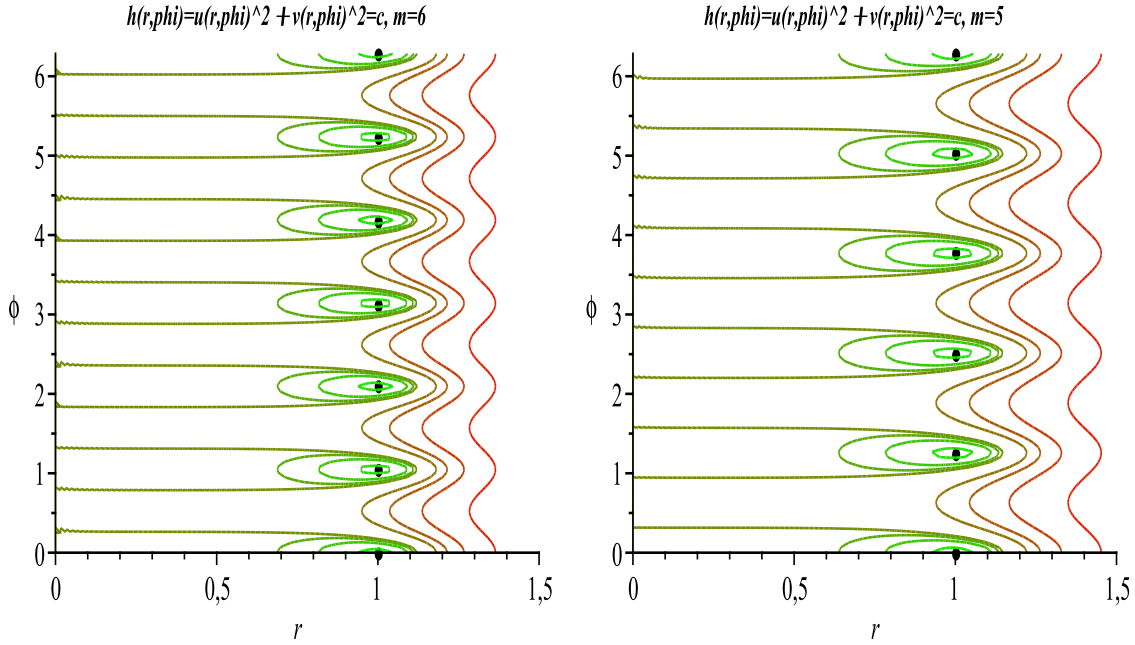


Abb. 3.12 $h(r,\varphi) = u(r,\varphi)^2 + v(r,\varphi)^2,\ m = 6, 5,$
Konturen $h(r,\varphi) = c$ mit `contours=[0,0.1,0.5,0.8,1,3,5,10,30]`

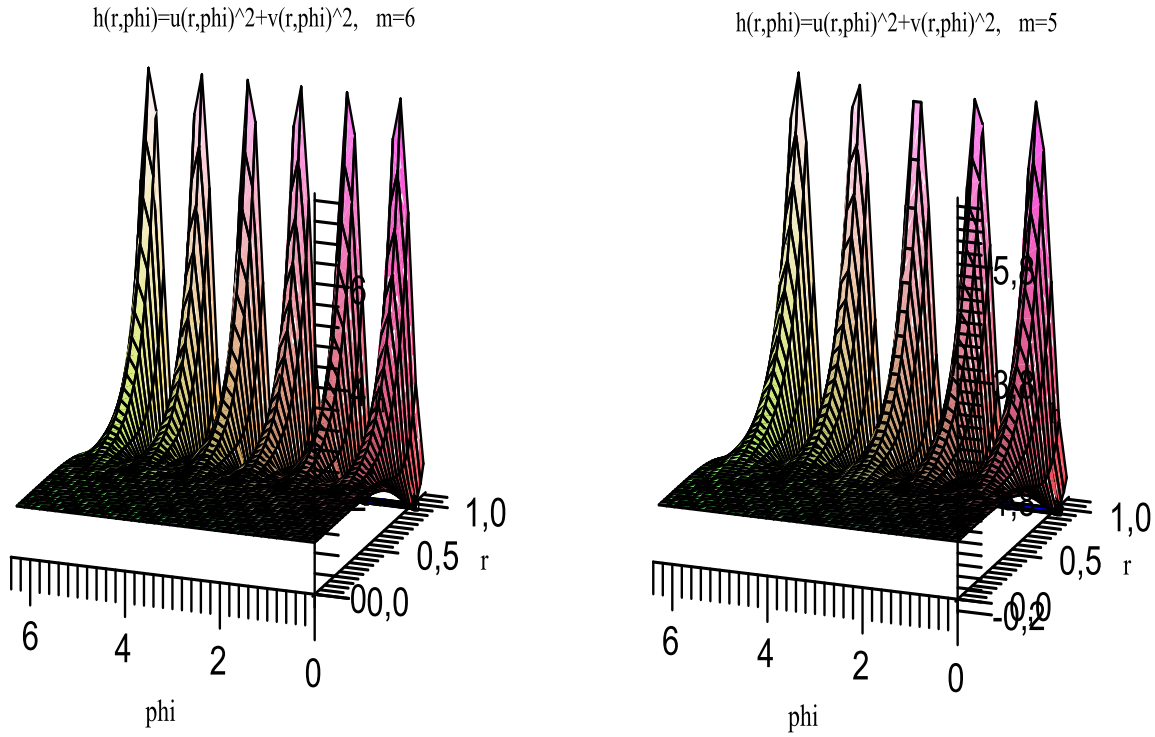


Abb. 3.13 $h(r,\varphi) = u(r,\varphi)^2 + v(r,\varphi)^2,\ m = 6, 5,$
mit Einheitswurzeln

Wir kehren zu den kartesischen Koordinaten zurück.
Dazu verwenden wir die Umrechnungsformeln zwischen Polar- und kartesischen Koordinaten aus Abschnitt 1.4 mit allen Besonderheiten und nehmen Bezug auf die Auswertungen der Funktion $f(z) = z^m - 1 = (x + \imath\, y)^m - 1$ mit ihrem Real- und Imaginärteil aus Abschnitt 3.1.
Die Koordinatenfunktionen seien $\tilde{u}(x, y)$ und $\tilde{v}(x, y)$, so dass sich das nichtlineare Gleichungssystem

$$\begin{aligned} \tilde{u}(x,y) &= \sqrt{x^2+y^2}^{\,m} \cos\left(m\left(C + \arctan\left(\frac{y}{x}\right)\right)\right) - 1 = 0, \\ \tilde{v}(x,y) &= \sqrt{x^2+y^2}^{\,m} \sin\left(m\left(C + \arctan\left(\frac{y}{x}\right)\right)\right) = 0, \end{aligned} \tag{3.9}$$

mit $a_u \le x \le a_o,\ b_u \le y \le b_o$ ergibt, wobei die Größe C die Sonderfälle in den Quadranten des Koordinatensystems berücksichtigen soll.
Weiterhin definieren wir die Funktionen und Gleichungen

$$\widetilde{F}(x,y) = (\widetilde{F}_1(x,y), \widetilde{F}_2(x,y))^T = (\tilde{u}(x,y)^2,\ \tilde{v}(x,y)^2)^T = (0,0)^T \tag{3.10}$$

sowie

$$\tilde{h}(x,y) = \tilde{u}(x,y)^2 + \tilde{v}(x,y)^2 \ge 0, \quad \tilde{h}(x,y) \to \min_{x,y}. \tag{3.11}$$

Die Funktion $\tilde{u}(x, y)$ entspricht dem Realteil $\Re(f(z))$, die Funktion $\tilde{v}(x, y)$ dem Imaginärteil $\Im(f(z))$. Man vergleiche die folgende gemeinsame Abbildung ihrer Konturen mit den Einzeldarstellungen in den Bildern 3.15 und 3.4.

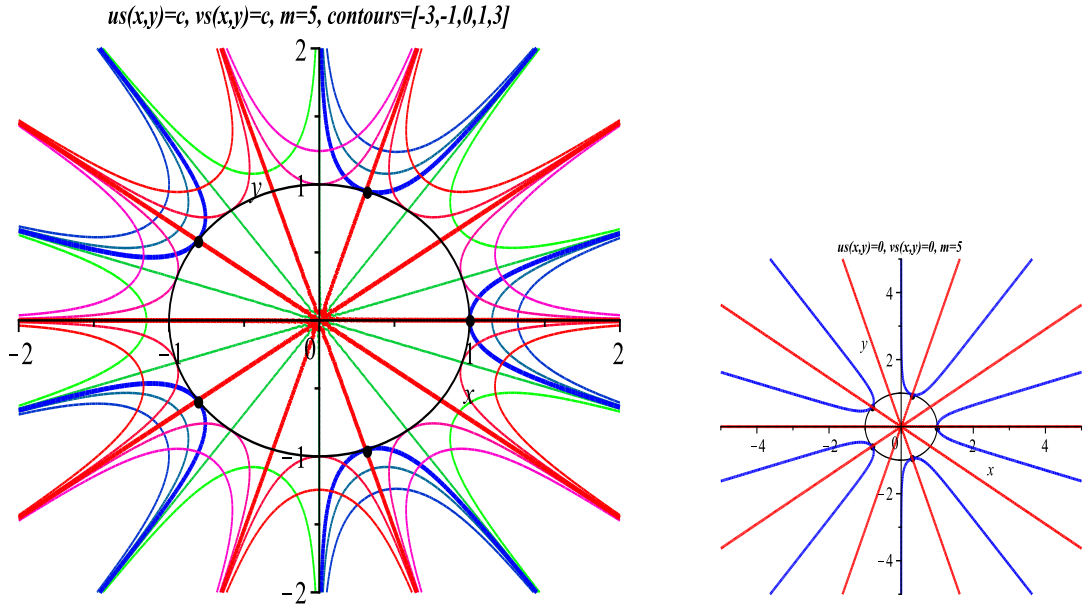


Abb. 3.14 Konturen $\tilde{u}(x, y) = c,\ \tilde{v}(x, y) = c,\ m = 5$,
mit `contours=[-3,-1,0,1,3]` (rechts nur $c = 0$) und Einheitswurzeln

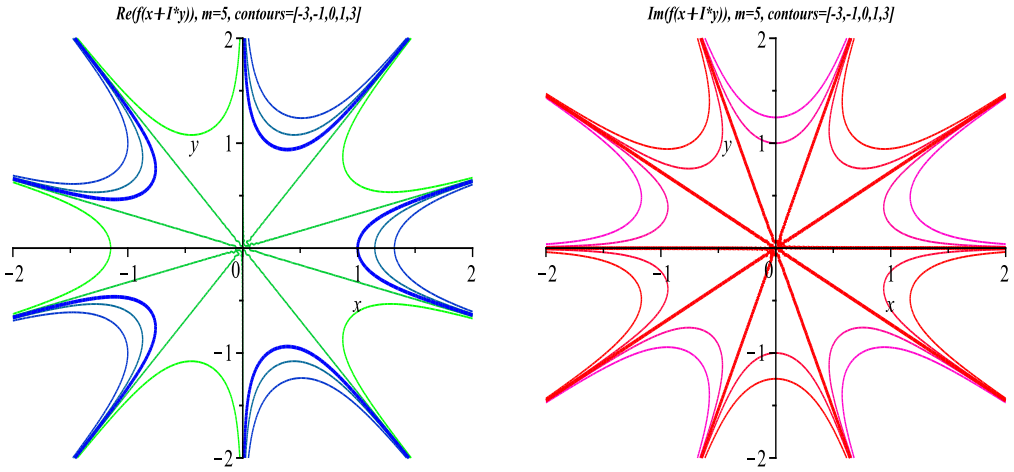


Abb. 3.15 $\tilde{u}(x,y) = \Re(f(z)) = c,\ \tilde{v}(x,y) = \Im(f(z)) = c,\ m = 5,$
Konturen mit `contours=[-3,-1,0,1,3]`

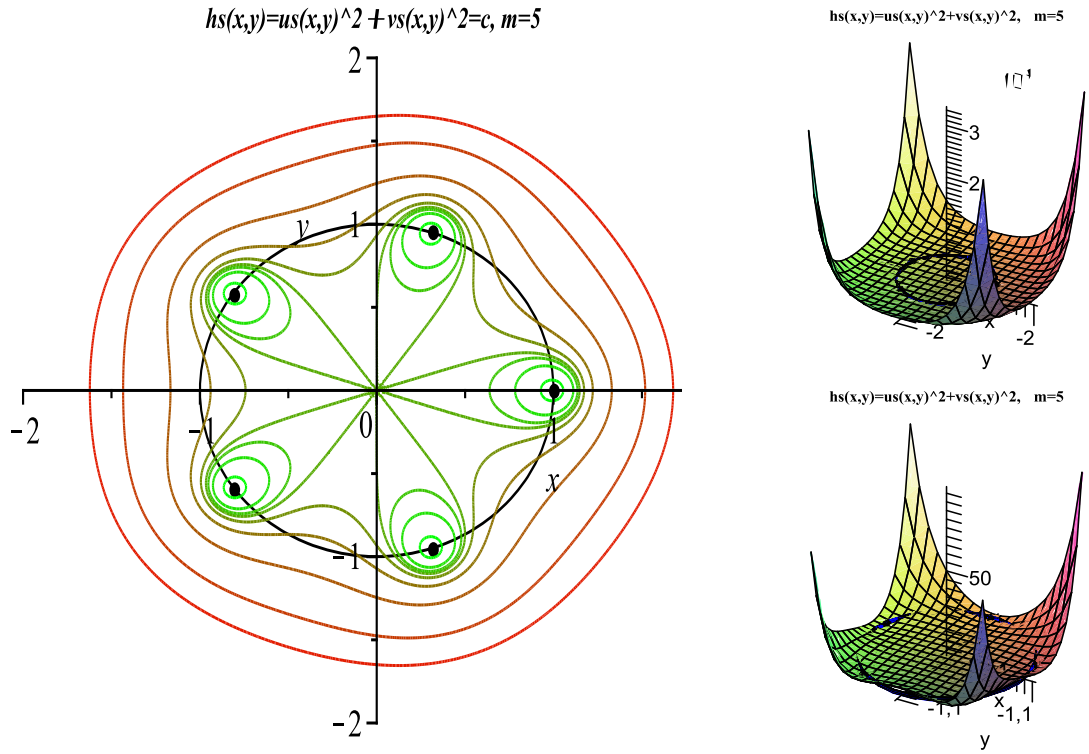


Abb. 3.16 links: Konturen $\tilde{h}(x,y) = c,\ m = 5,$
mit `contours=[0.1,0.5,0.8,1,1.5,3,10,50,150]`,
rechts: 3D-Graphik zu $\tilde{h}(x,y)$

In dieser Abbildung ist der qualitative Verlauf der berechneten Konturen der Funktion $\tilde{h}(x,y) = \tilde{u}(x,y)^2 + \tilde{v}(x,y)^2 = |f(z)|^2$ natürlich sehr ähnlich zu den Konturen von $|f(z)|$ in der Abb. 3.3.

Für die Implementation der Funktionen $\tilde{u}(x,y)$ und $\tilde{v}(x,y)$ in Maple betrachten wir 2 Varianten.
Zunächst notieren wir die Darstellung mit Verwendung der Bedingung `if then else`. Den Sonderfall $y = 0$ kann man explizit angeben, er wird aber auch durch die nachfolgenden Berechnung mit `arctan(y/x)` berücksichtigt (deshalb in den Prozeduren die Kennzeichnung von Zeilen als Kommentare).

```
> us:=proc(x,y)
   local r,phi;
   global m;
   r:=sqrt(x^2+y^2)^m;
   if x=0 then if y=0 then phi:=0
                      elif y>0 then phi:=Pi/2 else phi:=3*Pi/2 end if
        # elif y=0 then if x>0 then phi:=0 else phi:=Pi end if
                    else
                     phi:=arctan(y/x);
                     if x<0 then phi:=phi+Pi
                            elif y<0 then phi:=phi+2*Pi end if
   end if;
   r*cos(m*phi)-1;
  end proc:

> vs:=proc(x,y)
   local r,phi;
   global m;
   r:=sqrt(x^2+y^2)^m;
   if x=0 then if y=0 then phi:=0
                      elif y>0 then phi:=Pi/2 else phi:=3*Pi/2 end if
        # elif y=0 then if x>0 then phi:=0 else phi:=Pi end if
                    else
                     phi:=arctan(y/x);
                     if x<0 then phi:=phi+Pi
                            elif y<0 then phi:=phi+2*Pi end if
   end if;
   r*sin(m*phi);
  end proc:
```

In Abschnitt 1.4 wurde der Vorteil der `piecewise`-Version in Verbindung mit `And, Or` erläutert, auch bez. der Graphik. Deshalb bevorzugen wir hier die folgenden Funktionsdefinitionen.

```
> us:=unapply(piecewise(
       And(x=0,y=0),-1,
       And(x=0,y>0),y^m*cos(m*Pi/2)-1,
       And(x=0,y<0),(-y)^m*cos(m*3*Pi/2)-1,
       And(x>0,y=0),x^m-1,
       And(x<0,y=0),(-x)^m*cos(m*Pi)-1,
       x<0,(x^2+y^2)^(m/2)*cos(m*(Pi+arctan(y/x)))-1,
       x>0 and y<0,(x^2+y^2)^(m/2)*cos(m*(2*Pi+arctan(y/x)))-1,
       (x^2+y^2)^(m/2)*cos(m*arctan(y/x))-1),
       x,y):
> us(x,y);
```

$$\begin{cases} -1 & \mathrm{And}(x=0, y=0) \\ y^m \cos\left(\frac{1}{2} m\pi\right) - 1 & \mathrm{And}(x=0, 0<y) \\ (-y)^m \cos\left(\frac{3}{2} m\pi\right) - 1 & \mathrm{And}(x=0, y<0) \\ x^m - 1 & \mathrm{And}(0<x, y=0) \\ (-x)^m \cos(m\pi) - 1 & \mathrm{And}(x<0, y=0) \\ (x^2+y^2)^{\frac{1}{2}m} \cos\left(m\left(\pi + \arctan\left(\frac{y}{x}\right)\right)\right) - 1 & x<0 \\ (x^2+y^2)^{\frac{1}{2}m} \cos\left(m\left(2\pi + \arctan\left(\frac{y}{x}\right)\right)\right) - 1 & 0<x \text{ and } y<0 \\ (x^2+y^2)^{\frac{1}{2}m} \cos\left(m \arctan\left(\frac{y}{x}\right)\right) - 1 & otherwise \end{cases}$$

```
> vs:=unapply(piecewise(
      y=0,0,
      And(x=0,y>0),y^m*sin(m*Pi/2),
      And(x=0,y<0),(-y)^m*sin(m*3*Pi/2),
      x<0,(x^2+y^2)^(m/2)*sin(m*(Pi+arctan(y/x))),
      x>0 and y<0,(x^2+y^2)^(m/2)*sin(m*(2*Pi+arctan(y/x))),
      (x^2+y^2)^(m/2)*sin(m*arctan(y/x))),
      x,y):
> vs(x,y);
```

$$\begin{cases} 0 & y=0 \\ y^m \sin\left(\frac{1}{2} m\pi\right) & \mathrm{And}(x=0, 0<y) \\ (-y)^m \sin\left(\frac{3}{2} m\pi\right) & \mathrm{And}(x=0, y<0) \\ (x^2+y^2)^{\frac{1}{2}m} \sin\left(m\left(\pi + \arctan\left(\frac{y}{x}\right)\right)\right) & x<0 \\ (x^2+y^2)^{\frac{1}{2}m} \sin\left(m\left(2\pi + \arctan\left(\frac{y}{x}\right)\right)\right) & 0<x \text{ and } y<0 \\ (x^2+y^2)^{\frac{1}{2}m} \sin\left(m \arctan\left(\frac{y}{x}\right)\right) & otherwise \end{cases}$$

Abschließend noch Umformungen zur Betragsfunktion $|f(z)|$.

$$\begin{aligned} |f(z)| &= |z^m - 1| = |(x + \imath\, y)^m - 1| = |(r\, e^{\imath\,\varphi})^m - 1| \\ &= |r^m\, e^{\imath\, m\varphi} - 1| = |r^m \cos(m\varphi) - 1 + \imath\, r^m \sin(m\varphi)| \\ &= \sqrt{[r^m \cos(m\varphi) - 1]^2 + [r^m \sin(m\varphi)]^2} \\ &= \sqrt{r^{2m}[\cos^2(m\varphi) + \sin^2(m\varphi)] - 2r^m \cos(m\varphi) + 1} \qquad (3.12) \\ &= \sqrt{r^{2m} - 2r^m \cos(m\varphi) + 1} \\ &= \sqrt{[r^m - \cos(m\varphi)]^2 - \cos^2(m\varphi) + 1} \\ &= \sqrt{[r^m - \cos(m\varphi)]^2 + \sin^2(m\varphi)}. \end{aligned}$$

Die Funktion $|f(z)|$ hat ihre Nullstellen dort, wo

$$\begin{aligned} \sin(m\varphi) &= 0, \\ \cos(m\varphi) - r^m &= 0, \end{aligned} \tag{3.13}$$

ist. Das sind genau die Einheitswurzeln

$$r = 1, \quad \varphi_l = \frac{l}{m} 2\pi, \; l = 0, 1, 2, ..., m-1, \; \rightarrow \; \varphi_k = \frac{k}{m}\pi, \; k = 0, 2, 4, ..., 2m-2.$$

Maple-Funktionen zu $|f(z)|$ in Polar- und kartesischen Koordinaten und Anwendung für die nachfolgenden Abbildungen

```
> afz:=unapply(sqrt((r^m-cos(m*phi))^2+sin(m*phi)^2),r,phi);

> afxy:=proc(x,y)
   local r,phi;
   global m;
   r:=sqrt(x^2+y^2);
   phi:=piecewise(And(x=0,y=0),0,And(x=0,y>0),Pi/2,And(x=0,y<0),3*Pi/2,
                  And(x>0,y=0),0,And(x<0,y=0),Pi,
                  x<0,arctan(y/x)+Pi,
                  And(x>0,y<0),arctan(y/x)+2*Pi,
                  arctan(y/x));
   sqrt((r^m-cos(m*phi))^2+sin(m*phi)^2);
  end proc:
```

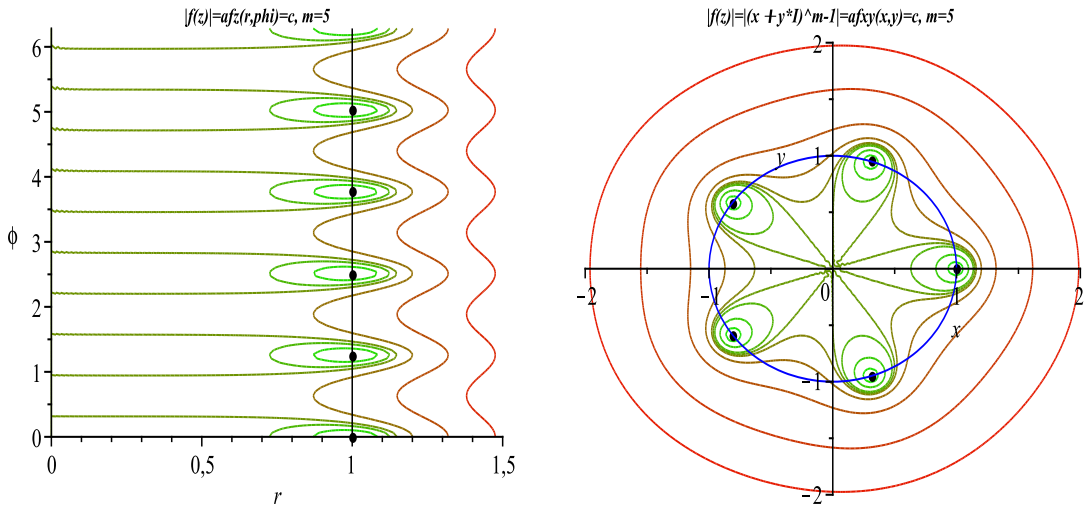


Abb. 3.17 links: Konturen $|f(z)| = \sqrt{[r^m - \cos(m\varphi)]^2 + \sin^2(m\varphi)} = c, \; m = 5,$ mit `contours=[0,0.5,0.8,1,1.5,3,6]`,
rechts: Konturen $|f(z)| = |(x + \imath\, y)^m - 1| = c, \; m = 5,$ mit Umrechnung $(x, y) \rightarrow (r, \varphi)$ und Anwendung von $\sqrt{[r^m - \cos(m\varphi)]^2 + \sin^2(m\varphi)},$ mit `contours=[0,0.3,0.6,0.9,1,1.1,1.5,3,10,30]`

Damit werden die Abbildungen 3.12, 3.16 und 3.3 bestätigt.

3.3 Wurzeln und kombinierte Lösungsverfahren

Nun sollen mehrere iterative Verfahren zur Lösung von Gleichungssystemen verknüpft werden. Dazu verwenden wir das bekannte Newton-Verfahren (NV) für zwei Ansätze sowie das in [9], [10] ausführlich beschriebene modifizierte Gradientenverfahren (MGV).
Alle diese Verfahren haben eine Iterationsvorschrift der Form

$$x^{(k+1)} = x^{(k)} + v^{(k)}, \quad k = 0, 1, 2, ..., k_{max}, \; x^{(0)} \text{ gegeben}, \tag{3.14}$$

wobei $v^{(k)}$ den "Verbesserungsvektor" im k-ten Schritt bezeichnen soll, also die Korrektur zu $x^{(k)}$.
Um nun die 3 Verfahren zu kombinieren, berechnen wir in jedem Iterationsschritt die Verbesserungsvektoren $v_i^{(k)}$, $i = 1, 2, 3$, der Verfahren und bestimmen mittels Konvexkombination den Verbesserungsvektor des kombinierten Verfahrens

$$v_{KOMB}^{(k)} = \lambda_1 v_1^{(k)} + \lambda_2 v_2^{(k)} + \lambda_3 v_3^{(k)}, \text{ wobei } \lambda_i \geq 0, \; i = 1, 2, 3, \; \sum_{i=1}^{3} \lambda_i^l = 1, \; l \geq 1. \tag{3.15}$$

Die Koeffizienten λ_i (Gewichte) und der Parameter l werden durch eine Skalierung vor Beginn der Iteration festgelegt.
Für das neue Verfahren erhalten wir

$$x^{(k+1)} = x^{(k)} + v_{KOMB}^{(k)}. \tag{3.16}$$

Es seien $f : \mathbb{R}^n \to \mathbb{R}^n$, $x \in \mathbb{R}^n$ und

$$f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{pmatrix}, \; F(x) = \begin{pmatrix} f_1^2(x) \\ \vdots \\ f_n^2(x) \end{pmatrix}, \tag{3.17}$$

$$h(x) = f_1^2(x) + f_2^2(x) + \ldots + f_n^2(x) \geq 0.$$

Mit x^* bezeichnen wir die Lösungen von $f(x) = 0$ bzw. $F(x) = 0$.
Damit ist $h(x)$ ein zu minimierendes Funktional, dessen Lösungen (Minimumstellen) ebenfalls x^* sind. Dafür ist $h(x^*) = 0$. An anderen lokalen Minima, die durchaus existieren können, gilt $h(x) > 0$.
Um eine Minimumstelle von $h(x)$ zu finden, verwenden wir ein Abstiegsverfahren, das sogenannte MGV. Dazu benötigen wir den Gradienten des Funktionals h an einer Stelle x

$$h'(x) = \nabla h(x) = \begin{pmatrix} h_{x_1}(x) \\ \vdots \\ h_{x_n}(x) \end{pmatrix}. \tag{3.18}$$

Anders als beim Gradientenverfahren folgt das MGV der Abstiegsgeraden an das Funktional nicht bis zu dem Punkt, an welchem das Funktional einen minimalen Wert annimmt, sondern hält (meist schon vorher, eventuell aber auch nachher), nämlich wenn die z-Koordinate der Abstiegsgeraden 0 wird. Dies hat eine (zum Teil wesentlich) kürzere oder zu lange "Schrittweite" als beim Gradientenverfahren zur Folge, dementsprechend mehr Schritte braucht man bis zur Lösung bzw. zur Minimumstelle. Allerdings macht dies das MGV auch "stabiler".

Für die Lösung des NLGS $f(x) = 0$ wollen wir also eine Kombination von NV für $f(x)$, NV für $F(x)$ und MGV für $h(x)$ benutzen.

Das NV für $f(x)$ lautet

$$v_{NV(f)}^{(k)} = -[Df(x^{(k)})]^{-1} f(x^{(k)}) \tag{3.19}$$

mit der Jacobi-Matrix der partiellen Ableitungen $Df(x) = \left(\frac{\partial f_i(x)}{\partial x_j}\right)_{i,j=1}^{n}$,

das NV für $F(x)$

$$v_{NV(F)}^{(k)} = -[DF(x^{(k)})]^{-1} F(x^{(k)}) \tag{3.20}$$

mit der Jacobi-Matrix $DF(x) = \left(\frac{\partial f_i^2(x)}{\partial x_j}\right)_{i,j=1}^{n}$,

und das MGV für $h(x)$

$$v_{MGV}^{(k)} = -\frac{h(x^{(k)})}{\sum\limits_{i=1}^{n} h_{x_i}^2(x^{(k)})}\, h'(x^{(k)}). \tag{3.21}$$

Mit der linearen konvexen Kombination

$$v_{KOMB}^{(k)} = \lambda_1 v_{NV(f)}^{(k)} + \lambda_2 v_{NV(F)}^{(k)} + \lambda_3 v_{MGV}^{(k)} \tag{3.22}$$

iterieren wir wie folgt

$$x^{(k+1)} = x^{(k)} + v_{KOMB}^{(k)}, \quad k = 0, 1, 2, ..., k_{max}. \tag{3.23}$$

Als zweite Abbruchbedingung wählen wir $h(x^{(k)}) = \|f(x^{(k)})\|_2^2 < \varepsilon$.

Für den Parametervektor λ benutzen wir den Hilfsvektor p und normieren diesen.

$$\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \frac{p}{\|p\|_l} \ (l = 1, 2) \ \text{ mit } \ p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}, \ p_i \geq 0 \ (i = 1, 2, 3). \tag{3.24}$$

In Bezug auf die Kombinationen zeigen sich einige interessante Effekte. So kann in einigen Fällen ein divergentes Einzelverfahren bewirken, dass auch jegliche Kombination mit konvergenten Verfahren nicht konvergiert. Ebenso ist zu beobachten, dass negative Einflüsse in der Kombination abgeschwächt werden, so dass diese dann konvergiert. Die Kombination konvergiert meist auch schneller als das langsamste Einzelverfahren.

Maple-Prozedur zum kombinierten Lösungsverfahren

In der Prozedur wurde bei der Implementation der einzelnen Methoden mehr Wert auf die Überschaubarkeit gelegt, denn auf ausgefeilte Arithmetik und Operationen. Der Rechenaufwand der Kombinationen ist natürlich höher als bei einem Einzelverfahren, bei drei Verfahren allerdings auch nicht dreimal so hoch, da viele Berechnungen gleichermaßen in allen Verfahren erfolgen müssen, so z. B. die Auswertung des Gleichungssystems in jedem Schritt.

Ergebnisse der Prozedur sind der letzte Näherungsvektor `xk` sowie die benötigte Iterationsanzahl `k`.

Einige Variablen stehen als globale Größen `global xv,ff,h;` zur Verfügung. In `xv` wird die Iterationsfolge $x^{(0)}, x^{(1)}, x^{(2)}, ..., x^{(k)}$ gespeichert zwecks nachträglicher graphischer Darstellungen, `ff` steht für Vektor der Funktionen $(f_1^2(x), f_2^2(x), ..., f_n^2(x))$ und `h` für die Funktion $h(x)$.

```
> kombn:=proc(f::procedure,n::posint,x0::vector,
              maxiter::posint,etol::numeric,para::vector,aus::name)

   local i,j,k,xk,hilf,df,dff,f_vec,ff_vec,df_mat,dff_mat,df_inv,dff_inv,
         uu,fh,fh1,xx,gamma,vec_nvf,vec_mgv,vec_nvff,vec_komb,
         hh,Dh,d1,dd;
   global xv,ff,h;

  fh1:=‘%+.16e‘;    # Ausgabeformate einstellen
  fh :=fh1;
  for k from 2 to n do fh:=cat(fh,‘ ‘,fh1); end do;

  xk:=evalm(x0):
  if aus=ja then
   printf(‘\nStartvektor        x = [‘||fh||‘]\n‘,seq(xk[i],i=1..n));
  end if;

  f_vec:=evalm(f(xk)):
  gamma:=evalm(transpose(f_vec)&*f_vec):
  xx:=matrix(n,0,[]):
  xv:=evalm(concat(xx,xk));

  if gamma<etol then RETURN(xk,0); end if;

  uu:=vector(n):
  ff:=vector(n): Dh:=vector(n):
  df:=matrix(n,n): dff:=matrix(n,n):

  for i from 1 to n do
   for j from 1 to n do
    hilf:=evalm(diff(f(uu)[i],uu[j]));
```

```
  df[i,j]:=unapply(hilf,uu);
 end do;
end do;
hh:=0:
for i from 1 to n do
 hilf:=evalf(evalm(f(uu)[i])^2);
 ff[i]:=unapply(evalm(hilf),uu);
 hh:=hh+evalm((ff[i](uu)))
end do;
h:=unapply(hh,uu);
for i from 1 to n do
 hilf:=evalm(diff(h(uu),uu[i]));
 Dh[i]:=unapply(hilf,uu);
end do;
for i from 1 to n do
 for j from 1 to n do
  hilf:=evalm(diff(ff[i](uu),uu[j]));
  dff[i,j]:=unapply(hilf,uu);
 end do;
end do;

df_mat:=matrix(n,n):  ff_vec:=vector(n):
dff_mat:=matrix(n,n): d1:=vector(n):

for k from 1 by 1 to maxiter do
 # Berechnung der Newton-Richtung fuer f(x)=0
 if evalm(para[1])<>0 then
  for i from 1 to n do
   for j from 1 to n do
    df_mat[i,j]:=evalm(df[i,j](xk));
   end do;
  end do;
  if det(df_mat)<>0 then
   df_inv:=evalm(inverse(df_mat));
   vec_nvf:=evalm(-df_inv&*f_vec);
  else
   vec_nvf:=vector(n,0);
   lprint(‘Jacobi-Matrix singulaer, setze Newton-Richtung [f(x)=0] Null‘):
  end if;
 else
  vec_nvf:=vector(n,0);
 end if;

 # Berechnung der Newton-Richtung fuer F(x)=0
 if evalm(para[2])<>0 then
  for i from 1 to n do ff_vec[i]:=evalm(ff[i](xk)) end do:
```

```
  for i from 1 to n do
   for j from 1 to n do
    dff_mat[i,j]:=evalm(dff[i,j](xk));
   end do;
  end do;
  if det(dff_mat)<>0 then
   dff_inv:=evalm(inverse(dff_mat));
   vec_nvff:=evalm(-dff_inv&*ff_vec);
  else
   vec_nvff:=vector(n,0);
   lprint(‘Jacobi-Matrix singulaer, setze Newton-Richtung [F(x)=0] Null‘):
  end if;
 else
  vec_nvff:=vector(n,0);
 end if;

 # Berechnung der modifizierten Gradienten-Richtung
 if evalm(para[3])<>0 then
  for i from 1 to n do d1[i]:=evalm(Dh[i](xk)) end do;
  dd:=evalm(transpose(d1)&*d1);
  if dd=0 then
   lprint(‘Abbruch wegen Nenner h’(x)=0‘):
   RETURN(xk,k-1);
  end if;
  vec_mgv:=evalm(-h(xk)/dd*d1);
  else
   vec_mgv:=vector(n,0);
 end if;

 vec_komb:=evalm(para[1]*vec_nvf+para[2]*vec_nvff+para[3]*vec_mgv);
 xk:=evalm(xk+vec_komb);
 xv:=evalm(concat(xv,xk));
 if aus=ja then
  printf(‘Iterationsvektor   x = [‘||fh||‘]\n‘,seq(xk[i],i=1..n));
 end if;
 f_vec:=evalm(f(xk)):
 gamma:=evalm(transpose(f_vec)&*f_vec);
 if gamma<etol then
  printf(‘Abbruch, da h(x^(k)) < epsilon‘);
  break;
 end if;
end do:

if k>maxiter then k:=k-1; end if;
[xk,k];
end proc:
```

3.3.1 Die Funktion von Himmelblau

Die Funktion von Himmelblau ist eine Testfunktion auf dem Gebiet der konvexen und diskreten Optimierung. Sie entsteht aus den nichtlinearen Gleichungssystemen

$$f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \begin{pmatrix} {x_1}^2 + x_2 - 11 \\ x_1 + {x_2}^2 - 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.25}$$

und damit

$$F(x) = \begin{pmatrix} f_1^2(x) \\ f_2^2(x) \end{pmatrix} = \begin{pmatrix} ({x_1}^2 + x_2 - 11)^2 \\ (x_1 + {x_2}^2 - 7)^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.26}$$

und hat die Gestalt eines multivariaten Polynoms

$$\begin{aligned} h(x) &= ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2 \\ &= x_1^4 + x_2^4 + 2(x_1^2 x_2 + x_1 x_2^2) - 21x_1^2 - 13x_2^2 - 14x_1 - 22x_2 + 170 \end{aligned} \tag{3.27}$$

mit

$$h'(x) = \begin{pmatrix} 4x_1^3 - 42x_1 + 4x_1x_2 + 2x_2^2 - 14 \\ 2x_1^2 + 4x_1x_2 + 4x_2^3 - 26x_2 - 22 \end{pmatrix}.$$

Die Funktion von Himmelblau hat 4 lokale Minima an den Stellen $x_1^* = (-2.805, 3.131)^T$, $x_2^* = (-3.799, -3.283)^T$, $x_3^* = (3.584, -1.848)^T$ und $x_4^* = (3, 2)^T$, die alle zugleich auch globale Minima mit Funktionswert 0 sind. Diese Minima sind die Lösungen des Gleichungssystems $f(x) = 0$.
Zudem hat die Funktion 4 Sattelpunkte und ein lokales Maximum an der Stelle $(-0.271, -0.923)^T$ (alle Werte auf 3 Nachkommastellen gerundet).

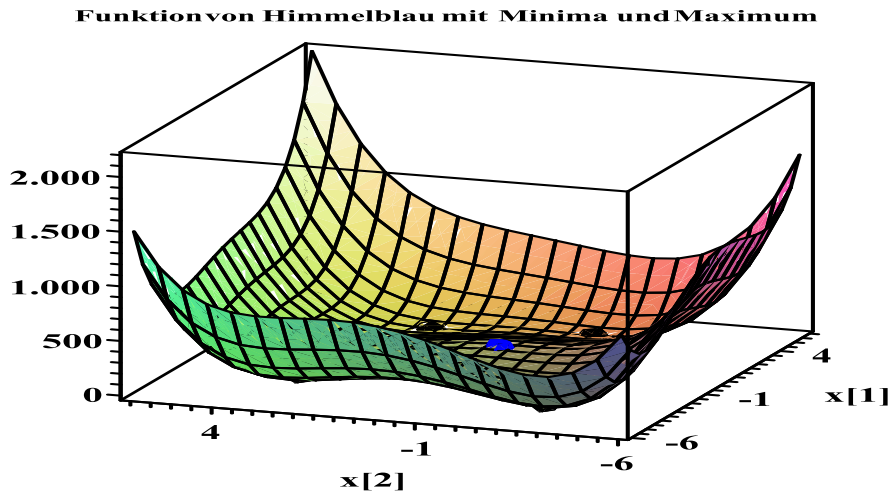


Abb. 3.18 Funktion von Himmelblau mit 4 Minima und 1 Maximum

Als Startvektor des kombinierten Lösungsverfahrens (3.23) wählen wir zunächst $x^{(0)} = (0,0)^T$, der Parametervektor sei $p = (0,1,1)$. Die Abbruchbedingung ist $h(x^{(k)}) = \|f(x^{(k)})\|_2^2 < \varepsilon = 10^{-16}$.

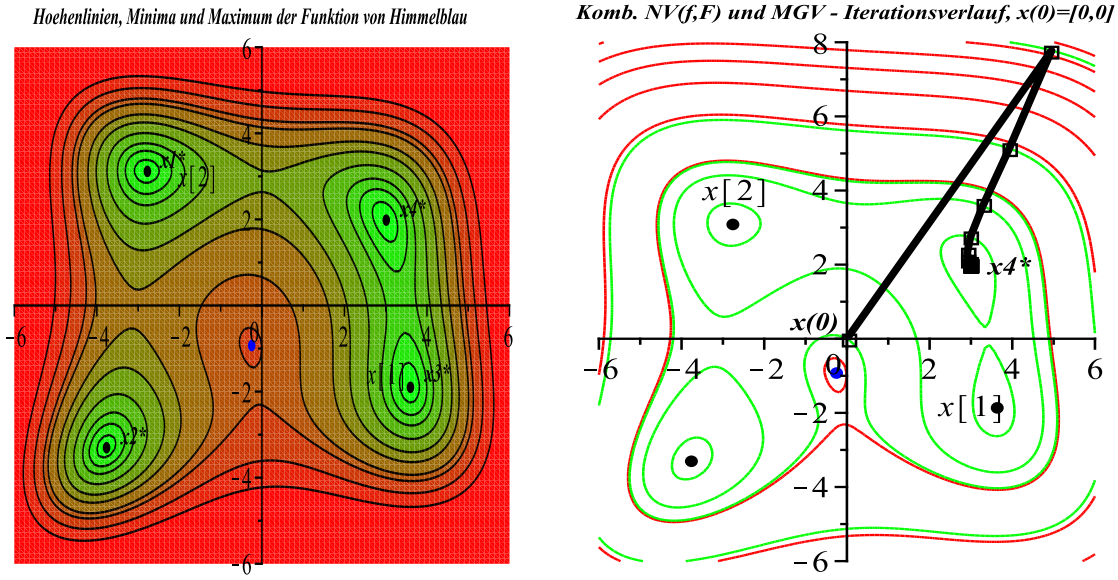


Abb. 3.19 Funktion von Himmelblau $h(x) = ({x_1}^2 + x_2 - 11)^2 + (x_1 + {x_2}^2 - 7)^2$,
links: Höhenlinien `contours=[3,10,20,40,70,100,150,180,250,350]`,
4 Minima und 1 Maximum,
rechts: konvergenter Iterationsverlauf mit der Kombination $p = (0,1,1)^T$
in 2-Norm und $x^{(0)} = (0,0)^T$ in 23 Schritten zu x_4^*

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	7 (x_4^*)
0	1	0	1, 2	33 (x_4^*)
0	0	1	1, 2	34 (x_4^*)
1	1	0	1	19 (x_4^*)
1	1	0	2	11 (x_4^*)
1	0	1	1	19 (x_4^*)
1	0	1	2	14 (x_4^*)
0	1	1	1	34 (x_4^*)
0	1	1	2	23 (x_4^*)
1	1	1	1	23 (x_4^*)
1	1	1	2	14 (x_4^*)
1	5	1	1	28 (x_4^*)
1	5	1	2	18 (x_4^*)

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	10 (x_4^*)
0	1	0	1, 2	41 (x_4^*)
0	0	1	1, 2	38 (x_1^*)
1	1	0	1	24 (x_4^*)
1	1	0	2	14 (x_4^*)
1	0	1	1	24 (x_3^*)
1	0	1	2	14 (x_3^*)
0	1	1	1	39 (x_3^*)
0	1	1	2	27 (x_4^*)
1	1	1	1	28 (x_4^*)
1	1	1	2	16 (x_3^*)
1	5	1	1	34 (x_3^*)
1	5	1	2	22 (x_4^*)

Tab. 3.1 Iteration von Kombinationen mit $x^{(0)} = (0,0)^T$ (links), $(0,-4)^T$ (rechts), mit Grenzvektor, $p = (p_1, p_2, p_3)$, $\varepsilon = 10^{-16}$

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	5 (x_2^*)
0	1	0	1, 2	30 (x_2^*)
0	0	1	1, 2	30 (x_2^*)
1	1	0	1	14 (x_2^*)
1	1	0	2	8 (x_2^*)
1	0	1	1	15 (x_2^*)
1	0	1	2	10 (x_2^*)
0	1	1	1	30 (x_2^*)
0	1	1	2	17 (x_2^*)
1	1	1	1	18 (x_2^*)
1	1	1	2	12 (x_2^*)
5	1	1	1	11 (x_2^*)
5	1	1	2	12 (x_2^*)
1	5,1	1,5	1	24 (x_2^*)
1	5,1	1,5	2	14 (x_2^*)

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	19 (x_2^*)
0	1	0	1, 2	63 (x_2^*)
0	0	1	1, 2	63 (x_2^*)
1	1	0	1	37 (x_2^*)
1	1	0	2	21 (x_2^*)
1	0	1	1	37 (x_2^*)
1	0	1	2	22 (x_2^*)
0	1	1	1	63 (x_2^*)
0	1	1	2	40 (x_2^*)
1	1	1	1	43 (x_2^*)
1	1	1	2	23 (x_2^*)
5	1	1	1	29 (x_2^*)
5	1	1	2	23 (x_2^*)
1	5,1	1,5	1	53 (x_2^*)
1	5,1	1,5	2	35 (x_2^*)

Tab. 3.2 Iteration von Kombinationen mit $x^{(0)} = (-2.5, -2)^T$ (links), $x^{(0)} = (-0.271, -0.923)^T$ (rechts), Grenzvektor, $p = (p_1, p_2, p_3)$, $\varepsilon = 10^{-16}$

Die Situationen in den Berechnungen ähneln sich. Kleine Abweichungen vom Startvektor $x^{(0)} = (-0.271, -0.923)^T$ können Folgen erzeugen, die zu einer anderen Lösung tendieren. Die Konvergenz ist jedoch praktisch für jeden Startvektor gegeben, es sei denn man beginnt exakt an der Maximumstelle. Bei Konvergenz ist der Grenzvektor eine der 4 Lösungen. Der polynomiale Charakter der Funktionen bewirkt das sehr gute lokale Konvergenzverhalten des NV(f), während die beiden anderen Methoden mehr Iterationen brauchen. Außerdem ist die Normierung des Parametervektors p in der 2-Norm, also $l = 2$ günstiger.

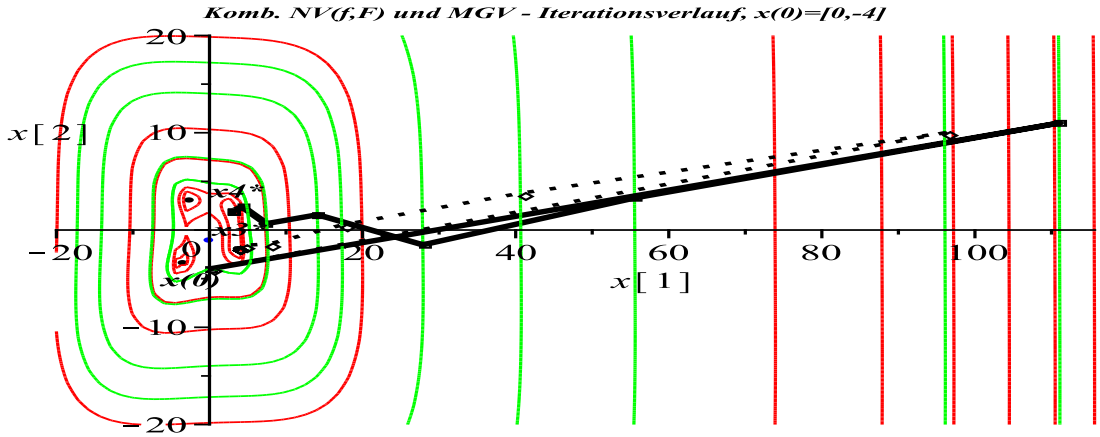


Abb. 3.20 Iterationsverläufe des kombinierten Verfahrens mit $x^{(0)} = (0, -4)^T$,
Linie durchgezogen: $p = (1, 0, 0)$ nach x_4^* in 10 Schritten,
Linie gepunktet: $p = (1, 1, 1)$ nach x_3^* in 16 Schritten mit 2-Norm

3.3.2 Die Einheitswurzeln

Eine Analogie zur Funktion von Himmelblau (3.27) findet man auch bei der Untersuchung der Einheitswurzeln.
Dazu betrachten wir die Beziehungen in (3.2) und können dort aus der Formel $f(z) = \Re(f(x + \imath\, y)) + \imath\, \Im(f(x + \imath\, y))$ die Gleichungssysteme

$$\begin{aligned} f_1(x,y) &= \Re(f(x+\imath\, y)) &= 0, \\ f_2(x,y) &= \Im(f(x+\imath\, y)) &= 0, \end{aligned} \tag{3.28}$$

$$\begin{aligned} f_1(x,y)^2 &= 0, \\ f_2(x,y)^2 &= 0, \end{aligned} \tag{3.29}$$

sowie das zu minimierende Funktional

$$h(x,y) = f_1(x,y)^2 + f_2(x,y)^2 = [\Re(f(x+\imath\, y))]^2 + [\Im(f(x+\imath\, y))]^2 \tag{3.30}$$

ableiten.
Ähnlich wie bei der Funktion von Himmelblau haben die Koordinatenfunktionen polynomialen Charakter. Das Funktional $h(x, y)$ ist ein multivariates Polynom und hat an den Stellen der Einheitswurzeln lokale Minima, die alle zugleich auch globale Minima mit Funktionswert 0 sind. Zudem hat die Funktion Sattelpunkte und ein lokales Maximum im Zentrum $(0, 0)^T$ mit Funktionswert 1.
Wir erkennen die Funktion auch in den Beziehungen $h(x, y) = |f(z)|^2$ und (3.11) wieder.
Wir testen die Gleichungssysteme (3.28) und (3.29) sowie das Funktional (3.30) in zwei Varianten. Dabei spielt die anfängliche Definition der Koordinatenfunktionen $f_1(x, y)$, $f_2(x, y)$ die entscheidende Rolle. Danach betrachten wir das System (3.9) - (3.11) als dritte Variante.
Es wird sich zeigen, dass der erste Versuch erfolglos ist. Dagegen führen dann die Varianten 2 und 3 zu denselben Ergebnissen. Dabei erhält man mit den komplexeren Formeln (3.9) - (3.11) etwas längere Rechenzeiten.

Implementationen in Maple

Zur schrittweisen Vorgehensweise erfolgen entsprechende Bemerkungen, die sowohl Möglichkeiten als auch auftretende Schwierigkeiten aufzeigen sollen.

Variante 1

(a) Zwecks Aufstellung des Gleichungssystems (3.28) untersuchen zunächst wir, ob wir aus der Funktion $f(x+\imath\, y) = (x+\imath\, y)^m - 1$ ihren Real- und Imaginärteil separieren, berechnen und weiterverwenden können. Dabei setzen schon wir voraus, dass die Variablen x, y selber reellwertig sind und m ganzzahlig positiv ist. Das Ergebnis ist kommentiert.

```
> assume(m::posint):
  assume(x,real):
  assume(y,real):
> # Real- und Imaginaerteil werden nicht ausgewertet bei allgemeinem m
  f:=unapply((x+y*I)^m-1,x,y);
                                # genauso
  f(x,x);                       # expand(f(x,y));
  Re(f(x,y)), Im(f(x,y));       # expand(Re(f(x,y))), expand(Im(f(x,y)));
                                # Re(expand(f(x,y))), Im(expand(f(x,y)));
```

$$f := (x\sim, y\sim) \to (x\sim + \mathrm{I}\, y\sim)^{m\sim} - 1$$
$$(x\sim + \mathrm{I}\, y\sim)^{m\sim} - 1$$
$$-1 + \Re\left((x\sim + \mathrm{I}\, y\sim)^{m\sim}\right),\ \Im\left((x\sim + \mathrm{I}\, y\sim)^{m\sim}\right)$$

(b) Wir verwenden nun einen konkreten Wert m, die Variablen x, y sind weiterhin reellwertig.

```
> m:=5:
  # x,y als reelle Groessen vereinbaren, sonst keine Auswertung
  assume(x,real):
  assume(y,real):
> f:=unapply((x+y*I)^m-1,x,y);
  f(x,y);
  expand(f(x,y));
  Re(f(x,y)), Im(f(x,y));
  expand(Re(f(x,y)));           # simplify(Re(f(x,y)));

  # richtige Reihenfolge, zuerst expand
  Re(expand(f(x,y))), Im(expand(f(x,y)));
```

$$f := (x\sim, y\sim) \to (x\sim + \mathrm{I}\, y\sim)^5 - 1$$
$$(x\sim + \mathrm{I}\, y\sim)^5 - 1$$
$$x\sim^5 + 5\mathrm{I}\, x\sim^4 y\sim - 10x\sim^3 y\sim^2 - 10\mathrm{I}\, x\sim^2 y\sim^3 + 5x\sim y\sim^4 + \mathrm{I}\, y\sim^5 - 1$$
$$-1 + \Re\left((x\sim + \mathrm{I}\, y\sim)^5\right),\ \Im\left((x\sim + \mathrm{I}\, y\sim)^5\right)$$
$$-1 + \Re\left((x\sim + \mathrm{I}\, y\sim)^5\right)$$
$$x\sim^5 - 10x\sim^3 y\sim^2 + 5x\sim y\sim^4 - 1,\ 5x\sim^4 y\sim - 10x\sim^2 y\sim^3 + y\sim^5$$

(c) Mit der Funktion `f` testen wir den Aufruf des Lösungsverfahrens mittels `erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja)`. Dazu definieren wir die vektorielle Funktion $f(\mathbf{x})$, was eine Umbenennung der bisherigen Funktion in `fk` notwendig macht.

```
> n:=2:
  fk:=unapply((x+y*I)^m-1,x,y);
  xx:=vector(n):                # Typ von xx nicht definiert,
                                # kann auch  komplexe Komponenten haben
> f:=xx->evalm([Re(fk(xx[1],xx[2])),Im(fk(xx[1],xx[2]))]);
  f(xx);
  f(xx)[1];
  f(xx)[2];
  x0:=vector(n,[0,0]);
  f(x0);
  f([1,1]);
```

$$fk := (x\sim, y\sim) \to (x\sim + \mathrm{I}\, y\sim)^5 - 1$$
$$f := xx \to evalm([\Re(fk(xx_1, xx_2)), \Im(fk(xx_1, xx_2))])$$
$$\left[-1 + \Re\left((xx_1 + \mathrm{I}\, xx_2)^5\right) \quad \Im\left((xx_1 + \mathrm{I}\, xx_2)^5\right)\right]$$
$$-1 + \Re\left((xx_1 + \mathrm{I}\, xx_2)^5\right)$$
$$\Im\left((xx_1 + \mathrm{I}\, xx_2)^5\right)$$
$$x0 := [0\ 0]$$
$$[-1\ 0]$$
$$[-5\ \ -4]$$

Machbar sind auch Aufrufe von `f` mit komplexen Argumenten wie `f([1,I]), f([0,I]), f([I,-1]), f([I,I])`.
Der Prozeduraufruf führt zu einer Fehlermeldung. Es gibt also Probleme mit dem Real- und Imaginaerteil sowie der interen Auswertung der Funktion `abs`.

```
> Digits:=20:
> x0:=vector(n,[1,1]);   # x0:=vector(n,[0,0]) ungeeigneter Startvektor
  mmax:=50:
  epsi:=1E-16:
  para_nvf_nvff_mgv:=vector(3,[1.0,1.0,1.0]):
  vec_para:= evalm((1/norm(para_nvf_nvff_mgv,1))*para_nvf_nvff_mgv);
  erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja);
```

$$x0 := [1\ 1]$$
$$vec\_para := [0.33333333333333333333\ 0.33333333333333333333\ 0.33333333333333333333]$$

```
Startvektor        x = [+1.0000000000000000e+00 +1.0000000000000000e+00]
Error, (in simpl/abs) abs is not differentiable at non-real arguments
```

Variante 2

(a) Unter Verwendung der Funktion `fk:=unapply((x+y*I)^m-1,x,y)` und des Befehls `subs` findet man eine lauffähige Version. Dabei erfolgt eine kleine Variablenumbenennung. Zuerst definieren wir die beiden Koordinaten `f1, f2` mit indizierten Variablen eines Vektors.

```
> m:=5:
  assume(x1,real):
  assume(x2,real):
  fk:=unapply((x1+x2*I)^m-1,x1,x2);
  fk(x1,x2);

> x:=vector(n):
  f1:=subs(x2=x[2],subs(x1=x[1],Re(expand(fk(x1,x2)))));
  f2:=subs(x2=x[2],subs(x1=x[1],Im(expand(fk(x1,x2)))));
```

$$fk := (x1\sim, x2\sim) \to (x1\sim + \mathrm{I}\, x2\sim)^5 - 1$$
$$(x1\sim + \mathrm{I}\, x2\sim)^5 - 1$$
$$f1 := x_1^5 - 10x_1^3x_2^2 + 5x_1x_2^4 - 1$$
$$f2 := 5x_1^4x_2 - 10x_1^2x_2^3 + x_2^5$$

(b) Aus den Koordinaten `f1, f2` konstruiert man die gesuchte Vektorfunktion `f`. Dabei ist die Verwendung der Befehls `unapply` dem Pfeiloperator vorzuziehen.

```
# Pfeiloperator zu "schwach"
f:=x->[f1,f2];
f(x);
x0:=vector(n,[0,0]);
f(x0);
```

$$f := x \to [f1, f2]$$
$$[x_1^5 - 10x_1^3x_2^2 + 5x_1x_2^4 - 1,\ 5x_1^4x_2 - 10x_1^2x_2^3 + x_2^5]$$
$$x0 := [0\ 0]$$
$$[x_1^5 - 10x_1^3x_2^2 + 5x_1x_2^4 - 1,\ 5x_1^4x_2 - 10x_1^2x_2^3 + x_2^5]$$

```
> # besser unapply
  f:=unapply([f1,f2],x);
  f(x);
  x0:=vector(n,[0,0]);
  f(x0);                          # korrekt
  f([1,1]);
  f([1,0]);
  f([0,1]);
```

$$f := x \to [x_1^5 - 10x_1^3x_2^2 + 5x_1x_2^4 - 1,\ 5x_1^4x_2 - 10x_1^2x_2^3 + x_2^5]$$
$$[x_1^5 - 10x_1^3x_2^2 + 5x_1x_2^4 - 1,\ 5x_1^4x_2 - 10x_1^2x_2^3 + x_2^5]$$
$$x0 := [0\ 0]$$
$$[-1\ 0]$$
$$[-5\ \ -4]$$
$$[0\ 0]$$
$$[-1\ 1]$$

(c) Mit der letzten Funktion `f:=unapply([f1,f2],x)` ist nun der Aufruf des Lösungsverfahrens mittels `erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja)` erfolgreich. Der Startvektor $(0,0)^T$ ist jedoch ungeeignet.

```
> x0:=vector(n,[0,0]);                    # singulaerer Punkt
  mmax:=50:
  epsi:=1E-16:
  para_nvf_nvff_mgv:=vector(3,[1.0,1.0,1.0]):
  vec_para:= evalm((1/norm(para_nvf_nvff_mgv,2))*para_nvf_nvff_mgv);
  erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja);
  evalm(erg[1]);
  erg[2];

> xv:=evalm(xv):
> cd:=coldim(xv);
```

$$x0 := [0\ 0]$$
$$vec\_para := [0.3333333333333333333\ 0.3333333333333333333\ 0.3333333333333333333]$$

```
Startvektor        x = [+0.0000000000000000e+00 +0.0000000000000000e+00]
‘Jacobi-Matrix singulaer, setze Newton-Richtung [f(x)=0] Null‘
‘Jacobi-Matrix singulaer, setze Newton-Richtung [F(x)=0] Null‘
‘Abbruch wegen Nenner h’(x)=0‘
```

$$erg := xk, 0$$
$$[0\ 0]$$
$$0$$
$$cd := 1$$

(d) Wir testen das kombinierte Lösungsverfahren mit anderen Startvektoren und machen dazu geeignete Auswertungen.

```
> x0:=vector(n,[1.2,0.6]);
  para_nvf_nvff_mgv:=vector(3,[1.0,1.0,1.0]):
  vec_para:= evalm((1/norm(para_nvf_nvff_mgv,1))*para_nvf_nvff_mgv);
  erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja);
  evalm(erg[1]);
  erg[2];

  xv1:=evalm(xv):
  cd1:=coldim(xv1);
```

$$x0 := [1.2\ 0.6]$$
$$vec\_para := [0.3333333333333333333\ 0.3333333333333333333\ 0.3333333333333333333]$$

```
Startvektor        x = [+1.2000000000000000e+00 +6.0000000000000000e-01]
Iterationsvektor   x = [+1.028477366255144 0e+00 +4.8049382716049383e-01]
Iterationsvektor   x = [+8.7717542125849050e-01 +3.3739592485574723e-01]
Iterationsvektor   x = [+7.7762371651616423e-01 +1.2239562318486033e-01]
Iterationsvektor   x = [+9.5563185394177182e-01 -9.6930514014249522e-02]
Iterationsvektor   x = [+9.7221752314260443e-01 -2.2384733252906328e-02]
Iterationsvektor   x = [+9.9103873704433 74e-01 -5.6918380263966536e-03]
Iterationsvektor   x = [+9.9707634523441790e-01 -1.7585023178452178e-03]
Iterationsvektor   x = [+9.9903271841805752e-01 -5.7220468344555774e-04]
Iterationsvektor   x = [+9.9967838363570560e-01 -1.8925514999707961e-04]
Iterationsvektor   x = [+9.9989288470117548e-01 -6.2922597915183070e-05]
Iterationsvektor   x = [+9.9996430491952104e-01 -2.0956220931809851e-05]
Iterationsvektor   x = [+9.9998810275313698e-01 -6.9834120308772361e-06]
Iterationsvektor   x = [+9.9999603437474743e-01 -2.3275824476210667e-06]
Iterationsvektor   x = [+9.9999867813866053e-01 -7.7583620142853063e-07]
Iterationsvektor   x = [+9.9999955938108070e-01 -2.5860933233889596e-07]
Iterationsvektor   x = [+9.9999985312719659e-01 -8.6202806917504353e-08]
Iterationsvektor   x = [+9.9999995104241772e-01 -2.8734235210227179e-08]
Iterationsvektor   x = [+9.9999998368080800e-01 -9.5780746520520797e-09]
Iterationsvektor   x = [+9.9999999456026957e-01 -3.1926911338668372e-09]
Iterationsvektor   x = [+9.9999999818675655e-01 -1.0642303316426007e-09]
Iterationsvektor   x = [+9.9999999939558552e-01 -3.5474343873497707e-10]
Abbruch, da h(x^(k)) < epsilon
```

$$erg := [xk, 21]$$
$$[0.99999999939558551887\ -3.5474343873497707269\,10^{-10}]$$
$$21$$
$$cd1 := 22$$

```
> x0:=vector(n,[1.2,0.6]);
  para_nvf_nvff_mgv:=vector(3,[1.0,1.0,1.0]):
  vec_para:= evalm((1/norm(para_nvf_nvff_mgv,2))*para_nvf_nvff_mgv);
  erg:=kombn(f,n,x0,mmax,epsi,vec_para,ja);
  evalm(erg[1]);
  erg[2];
```

```
xv2:=evalm(xv):
cd2:=coldim(xv2);
```

$$x0 := [1.2\ 0.6]$$

$$vec\_para := [0.57735026918962576452\ 0.57735026918962576452\ 0.57735026918962576452]$$

```
Startvektor        x = [+1.2000000000000000e+00 +6.0000000000000000e-01]
Iterationsvektor   x = [+9.0291408370588146e-01 +3.9300923682386750e-01]
Iterationsvektor   x = [+6.7688823720990452e-01 +5.7279670706973164e-02]
Iterationsvektor   x = [+1.5438299870677283e+00 -3.1524955899481308e-01]
Iterationsvektor   x = [+1.2132451917448649e+00 -2.1542160646622426e-01]
Iterationsvektor   x = [+1.0094856440103479e+00 -1.0091745594908788e-01]
Iterationsvektor   x = [+9.7727112047800360e-01 +6.9918513425308358e-03]
Iterationsvektor   x = [+1.0046361090579789e+00 -1.8666244911470517e-03]
Iterationsvektor   x = [+9.9932414574254755e-01 +2.4931816072139199e-04]
Iterationsvektor   x = [+1.0001054671961549e+00 -3.9349443097133204e-05]
Iterationsvektor   x = [+9.9998370627721961e-01 +6.0682174369066328e-06]
Iterationsvektor   x = [+1.0000025211757706e+00 -9.3921320454718415e-07]
Iterationsvektor   x = [+9.9999960998539304e-01 +1.4528585151397772e-07]
Iterationsvektor   x = [+1.0000000603357722e+00 -2.2476061166369861e-08]
Iterationsvektor   x = [+9.9999999066603080e-01 +3.4770524994773578e-09]
Iterationsvektor   x = [+1.0000000014439702e+00 -5.3790204354391610e-10]
Abbruch, da h(x^(k)) < epsilon
```

$$erg := [xk, 15]$$

$$[1.0000000014439702344\ -5.379020435439160976\,10^{-10}]$$

$$15$$

$$cd2 := 16$$

(e) Das Funktional $h(x)$ ist eine globale Groesse aus der Prozedur `kombn`.
Wir brauchen sie auch für die graphischen Auswertungen und Darstellungen.
Von den folgenden Definitionen nehmen wir die letzte.

```
> v:=vector(n,[x,y]):
  hh:=h(v);
> x:=vector(n,[x1,x2]):
  hh:=h(x);
> # guenstige Definition der zu minimierenden Funktion
  x:=vector(n,[]):
  hh:=h(x);
```

$$hh := \left(x_1^5 - 10.x_1^3x_2^2 + 5.x_1x_2^4 - 1.\right)^2 + \left(5.x_1^4x_2 - 10.x_1^2x_2^3 + x_2^5\right)^2$$

(f) Graphische Darstellungen zur Funktion $h(x)$, zu ihren Höhenlinien, zu den beiden Iterationsverläufen aus dem Punkt (d) zusammen mit den Einheitswurzeln.
Es sollen auch die zugehörigen Maple-Anweisungen notiert werden.

```
> # 3D-Graph von h(x)
  pl1:=plot3d(hh,x[1]=-1.3..1.3,x[2]=-1.3..1.3,grid=[27,27],
                  axes=normal,orientation=[110,60]):
  pl2:=pointplot3d([seq([ns[i,1],ns[i,2],0],i=1..m)],color=black,
                  symbol=solidcircle,symbolsize=30):
  pl3:=implicitplot3d(x^2+y^2=1,x=-1..1,y=-1..1,z=-1..3,color=blue):
  display(pl1,pl2,pl3,title=`h(x), Einheitswurzeln`,
                  titlefont=[TIMES,BOLD,6],font=[TIMES,BOLD,6]);
```

```
> # Hoehenlinien von h(x)
  pl4:=contourplot(hh,x[1]=-1.3..1.3,x[2]=-1.3..1.3,filled=true,
                   grid=[100,100],coloring=[green,red],
                   contours=[0.3,0.6,0.9,0.99,1,1.01,1.1,1.5,2,3,6]):
  pl5:=pointplot([seq([ns[i,1],ns[i,2]],i=1..m)],color=black,
                   symbol=solidcircle,symbolsize=16):
  pl6:=implicitplot(x^2+y^2=1,x=-1..1,y=-1..1,color=blue):
  display(pl4,pl5,pl6,title=‘Hoehenlinien von h(x), Einheitswurzeln‘,
                   titlefont=[TIMES,BOLD,8]);

> # Iterationsverlauf und Hoehenlinien von h(x)
  pl7:=contourplot(hh,x[1]=-1.5..1.5,x[2]=-1.5..1.5,scaling=constrained,
       contours=[0.5,0.9,1,1.5,3,6],grid=[100,100],thickness=1,color=red):
  pl8:=contourplot(hh,x[1]=-1.5..1.5,x[2]=-1.5..1.5,
                   contours=[seq(h([xv1[1,i],xv1[2,i]]),i=1..5)],
                   grid=[100,100],thickness=1,color=gray):
  pl9:=plot([seq([xv1[1,i],xv1[2,i]],i=1..cd1)],style=line,
                   thickness=2,color=black):
  pl10:=pointplot([seq([xv1[1,i],xv1[2,i]],i=1..cd1)],
                   symbol=box,symbolsize=16,color=black):
  pl8a:=contourplot(hh,x[1]=-1.5..1.5,x[2]=-1.5..1.5,
                   contours=[seq(h([xv2[1,i],xv2[2,i]]),i=1..5)],
                   grid=[100,100],thickness=1,color=cyan):
  pl9a:=plot([seq([xv2[1,i],xv2[2,i]],i=1..cd2)],style=line,
                   thickness=1,color=green):
  pl10a:=pointplot([seq([xv2[1,i],xv2[2,i]],i=1..cd2)],
                   symbol=diamond,symbolsize=16,color=green):
  pl11:=textplot([[1.2,0.7,‘x(0)‘],[1.0,0.1,‘x*‘]],font=[TIMES,BOLD,9]):
  plots[display](pl6,pl7,pl8,pl9,pl10,pl11,pl8a,pl9a,pl10a,pl5,
      title=‘Komb. NV(f,F) und MGV - Iterationsverlauf, x(0)=[1.2,0.6]‘,
      titlefont=[TIMES,BOLD,7]);
```

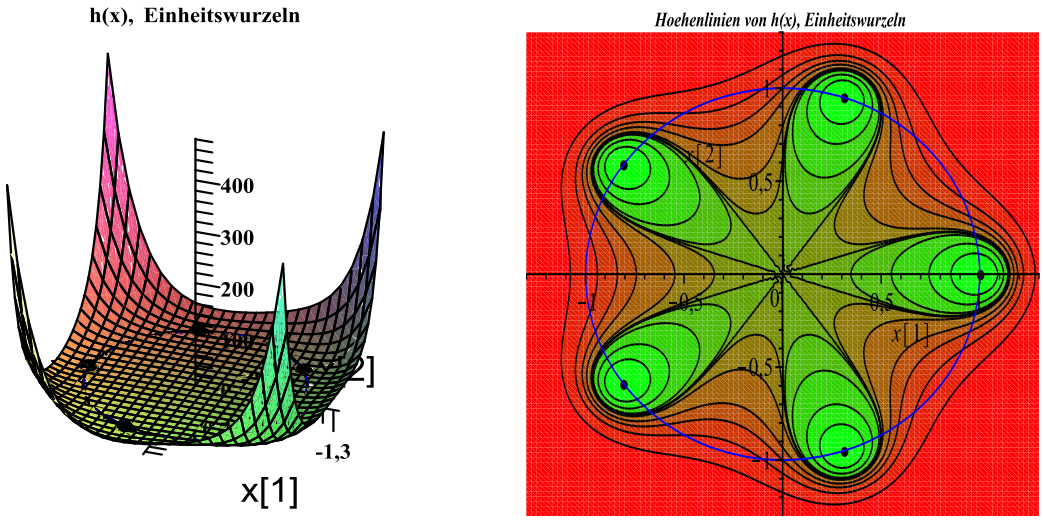


Abb. 3.21 $h(x)$ mit Einheitswurzeln, $m = 5$,
links: 3D-Graph, rechts: Höhenlinien
mit `contours=[0.3,0.6,0.9,0.99,1,1.01,1.1,1.5,2,3,6]`

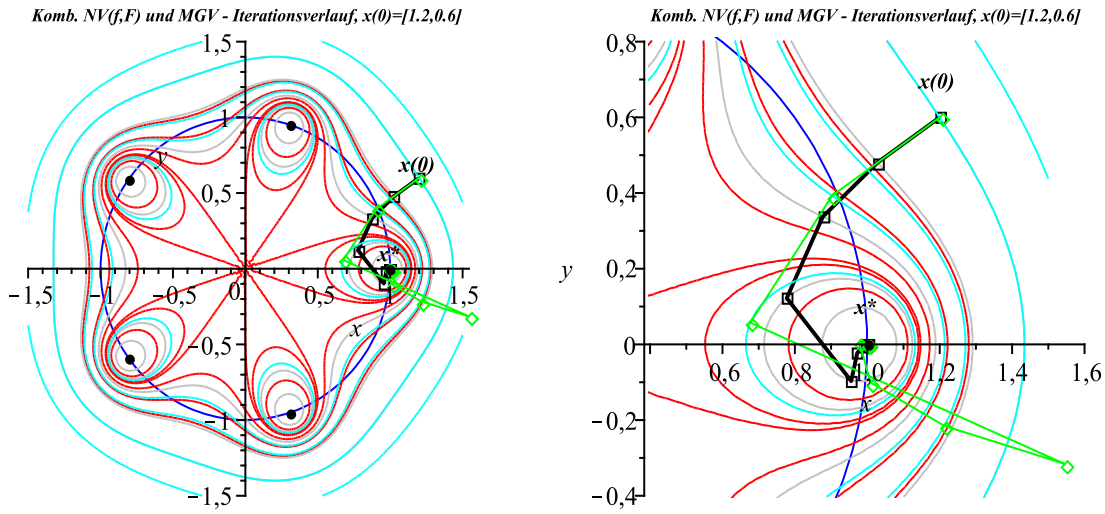


Abb. 3.22 Iterationsverläufe des kombinierten Verfahrens mit $x^{(0)} = (1.2, 0.6)^T$, $p = (1, 1, 1)$, Normen $l = 1, 2$ bei 21 (schwarz) bzw. 15 Schritten, rechts: Ausschnittsvergrößerung (Iteration siehe Punkt (d))

Variante 3

Hier verwenden wir die Beziehungen (3.9) - (3.11), also die komplizierteren Funktionen $\tilde{u}(x,y)^2$, $\tilde{v}(x,y)^2$ und das Funktional $\tilde{h}(x,y) = \tilde{u}(x,y)^2 + \tilde{v}(x,y)^2$.
Die Ergebnisse sind analog zu denen in Variante 2. Aber wegen der Komplexität und zusätzlichen Umrechnungen zwischen den kartesischen und Polarkoordinaten brauchen die Berechnungen mehr Computerzeit.
Wir notieren nur den Anfang der Anweisungen und überlassen dem Leser in Analogie zur Variante 2 weitere Untersuchungen und Auswertungen.

```
> m:='m':
> us:=unapply(piecewise(
      And(x=0,y=0),-1,
      And(x=0,y>0),y^m*cos(m*Pi/2)-1,
      And(x=0,y<0),(-y)^m*cos(m*3*Pi/2)-1,
      And(x>0,y=0),x^m-1,
      And(x<0,y=0),(-x)^m*cos(m*Pi)-1,
      x<0,(x^2+y^2)^(m/2)*cos(m*(Pi+arctan(y/x)))-1,
      x>0 and y<0,(x^2+y^2)^(m/2)*cos(m*(2*Pi+arctan(y/x)))-1,
      (x^2+y^2)^(m/2)*cos(m*arctan(y/x))-1),
      x,y):
> vs:=unapply(piecewise(
      y=0,0,
      And(x=0,y>0),y^m*sin(m*Pi/2),
      And(x=0,y<0),(-y)^m*sin(m*3*Pi/2),
      x<0,(x^2+y^2)^(m/2)*sin(m*(Pi+arctan(y/x))),
      x>0 and y<0,(x^2+y^2)^(m/2)*sin(m*(2*Pi+arctan(y/x))),
      (x^2+y^2)^(m/2)*sin(m*arctan(y/x))),
      x,y):
```

```
> m:=5:
  us(0,y);
  vs(0,y);
```

```
Error, (in us) numeric exception: division by zero
Error, (in vs) numeric exception: division by zero
```

```
> us(0,y) assuming y::positive;
  vs(0,y) assuming y::positive;
```

$$y^{-1}$$

$$y^{-5}$$

```
> us(x,0);
  vs(x,0);
```

$$\begin{cases} -1 & x = 0 \\ x^5 - 1 & 0 < x \\ x^5 - 1 & x < 0 \\ -\left(x^2\right)^{5/2} - 1 & x < 0 \\ \left(x^2\right)^{5/2} - 1 & otherwise \end{cases}$$

$$0$$

```
> n:=2:
  x:=vector(n):
  f:=x->evalm([us(x[1],x[2]),vs(x[1],x[2])]);
  f(x);

  x0:=vector(n,[0.0,0.0]);
  f(x0);
  f([1,1]);
  f([1,0]);
  f([0,1]);
```

$$f := x \to evalm([us(x_1, x_2),\ vs(x_1, x_2)])$$

$$\left[\begin{cases} -1 & \mathrm{And}(x_1 = 0, x_2 = 0) \\ -1 & \mathrm{And}(x_1 = 0, 0 < x_2) \\ -1 & \mathrm{And}(x_1 = 0, x_2 < 0) \\ x_1^5 - 1 & \mathrm{And}(0 < x_1, x_2 = 0) \\ x_1^5 - 1 & \mathrm{And}(x_1 < 0, x_2 = 0) \\ -(x_1^2 + x_2^2)^{5/2} \cos\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) - 1 & x_1 < 0 \\ (x_1^2 + x_2^2)^{5/2} \cos\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) - 1 & 0 < x_1 \text{ and } x_2 < 0 \\ (x_1^2 + x_2^2)^{5/2} \cos\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) - 1 & otherwise \end{cases}, \begin{cases} 0 & x_2 = 0 \\ x_2^5 & \mathrm{And}(x_1 = 0, 0 < x_2) \\ x_2^5 & \mathrm{And}(x_1 = 0, x_2 < 0) \\ -(x_1^2 + x_2^2)^{5/2} \sin\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) & x_1 < 0 \\ (x_1^2 + x_2^2)^{5/2} \sin\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) & 0 < x_1 \text{ and } x_2 < 0 \\ (x_1^2 + x_2^2)^{5/2} \sin\left(5 \arctan\left(\frac{x_2}{x_1}\right)\right) & otherwise \end{cases}\right]$$

$$x0 := [0.\ 0.]$$

$$[-1\ 0]$$

$$[-5\ -4]$$

$$[0\ 0]$$

$$[-1\ 1]$$

Zur Variante 2 wollen wir abschließend noch einige Testrechnungen machen und interessante Iterationsverläufe zeigen.
In Abschnitt 3.1 haben wir die Einzugsbereiche der Einheitswurzeln untersucht. Dabei stellten wir fest, dass die Grenzen dieser keine Geraden sind, sondern fractale Gebilde. Nimmt man nun Startpunkte aus diesen Bereichen, so können ganz unterschiedliche Iterationsverläufe entstehen. Auf jeden Fall tendieren die Folgen gegen eine der Einheitswurzeln

$$\omega_k = e^{\imath\, x_k} = e^{\imath\, 2\pi k/m} = \cos(\tfrac{2\pi k}{m}) + \imath\, \sin(\tfrac{2\pi k}{m}),\ k = 0, 1, 2, ..., m-1,\ m = 5. \quad (3.31)$$

Mit einigen Startpunkten sowie verschiedenen Parameterkonstellationen wollen wir die Iterationsfolgen berechnen und graphisch darstellen.

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	10 (ω_0)
0	1	0	1, 2	32 (ω_0)
0	0	1	1, 2	32 (ω_0)
1	1	0	1	18 (ω_0)
1	1	0	2	13 (ω_0)
1	0	1	1	18 (ω_0)
1	0	1	2	13 (ω_0)
0	1	1	1	32 (ω_0)
0	1	1	2	20 (ω_0)
1	1	1	1	20 (ω_0)
1	1	1	2	16 (ω_0)

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	10 (ω_4)
0	1	0	1, 2	27 (ω_4)
0	0	1	1, 2	27 (ω_4)
1	1	0	1	16 (ω_4)
1	1	0	2	14 (ω_4)
1	0	1	1	16 (ω_4)
1	0	1	2	14 (ω_4)
0	1	1	1	27 (ω_4)
0	1	1	2	18 (ω_4)
1	1	1	1	19 (ω_4)
1	1	1	2	18 (ω_4)

Tab. 3.3 Iteration von Kombinationen und Grenzvektor, $p = (p_1, p_2, p_3)$, $\varepsilon = 10^{-16}$, links: $x^{(0)} = (1.5, 0.75)^T$, rechts: $x^{(0)} = (0.5, 0.25)^T$

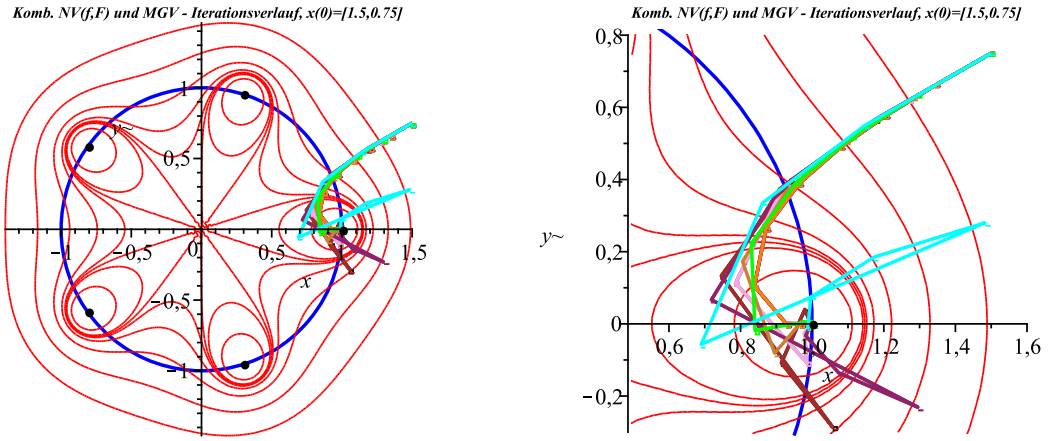


Abb. 3.23 Kombiniertes Lösungsverfahren mit $x^{(0)} = (1.5, 0.75)^T$, Iterationsverläufe, Einheitswurzeln, Höhenlinien `contours=[0.5,0.9,1,1.1,1.5,3,10,40]`, rechts: Ausschnittsvergrößerung

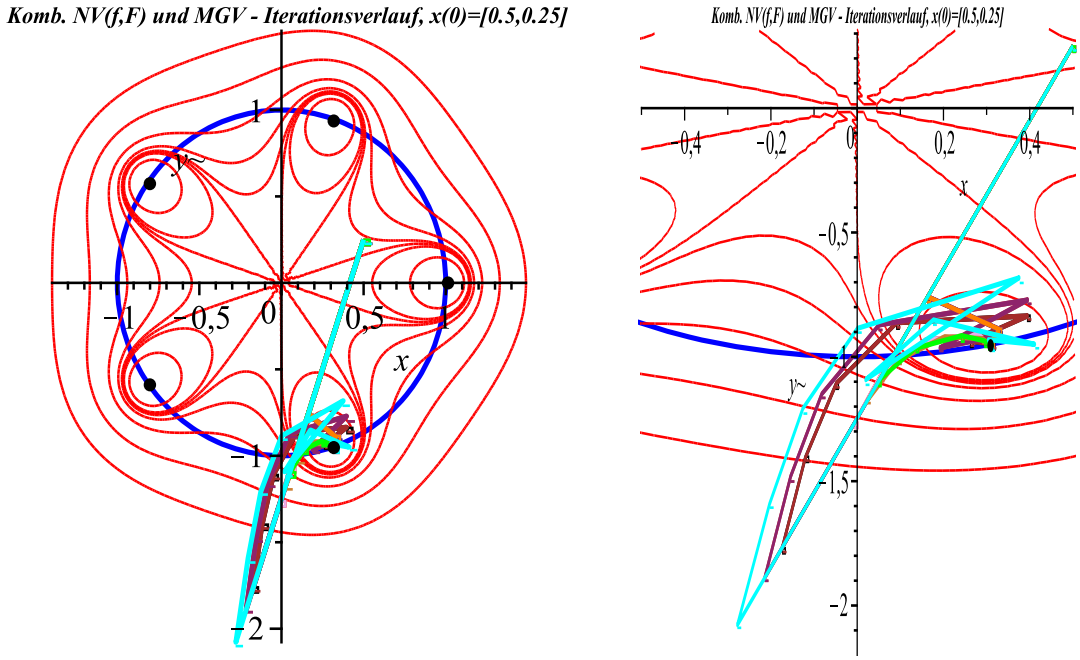


Abb. 3.24 Kombiniertes Lösungsverfahren mit $x^{(0)} = (0.5, 0.25)^T$, Iterationsverläufe, Einheitswurzeln, Höhenlinien `contours=[0.5,0.9,1,1.1,1.5,3,10,40]`, rechts: verzerrte Ausschnittsvergrößerung

p_1 NV(f)	p_2 NV(F)	p_3 MGV	l-Norm von p	Iterat.-anzahl
1	0	0	1, 2	18 (ω_4)
0	1	0	1, 2	64 (ω_2)
0	0	1	1, 2	64 (ω_2)
1	1	0	1	25 (ω_4)
1	1	0	2	20 (ω_4)
1	0	1	1	25 (ω_4)
1	0	1	2	20 (ω_4)
0	1	1	1	64 (ω_2)
0	1	1	2	25 (ω_3)
1	1	1	1	33 (ω_3)
1	1	1	2	23 (ω_4)
5	1	1	1	27 (ω_4)
5	1	1	2	23 (ω_4)
1	5	1	1	43 (ω_4)
1	5	1	2	25 (ω_2)
1	1	5	1	43 (ω_4)
1	1	5	2	25 (ω_2)

Tab. 3.4 Iteration von Kombinationen mit $x^{(0)} = (0.75, 0.5)^T$, mit Grenzvektoren $\omega_{2,3,4}$, $p = (p_1, p_2, p_3)$, $\varepsilon = 10^{-16}$

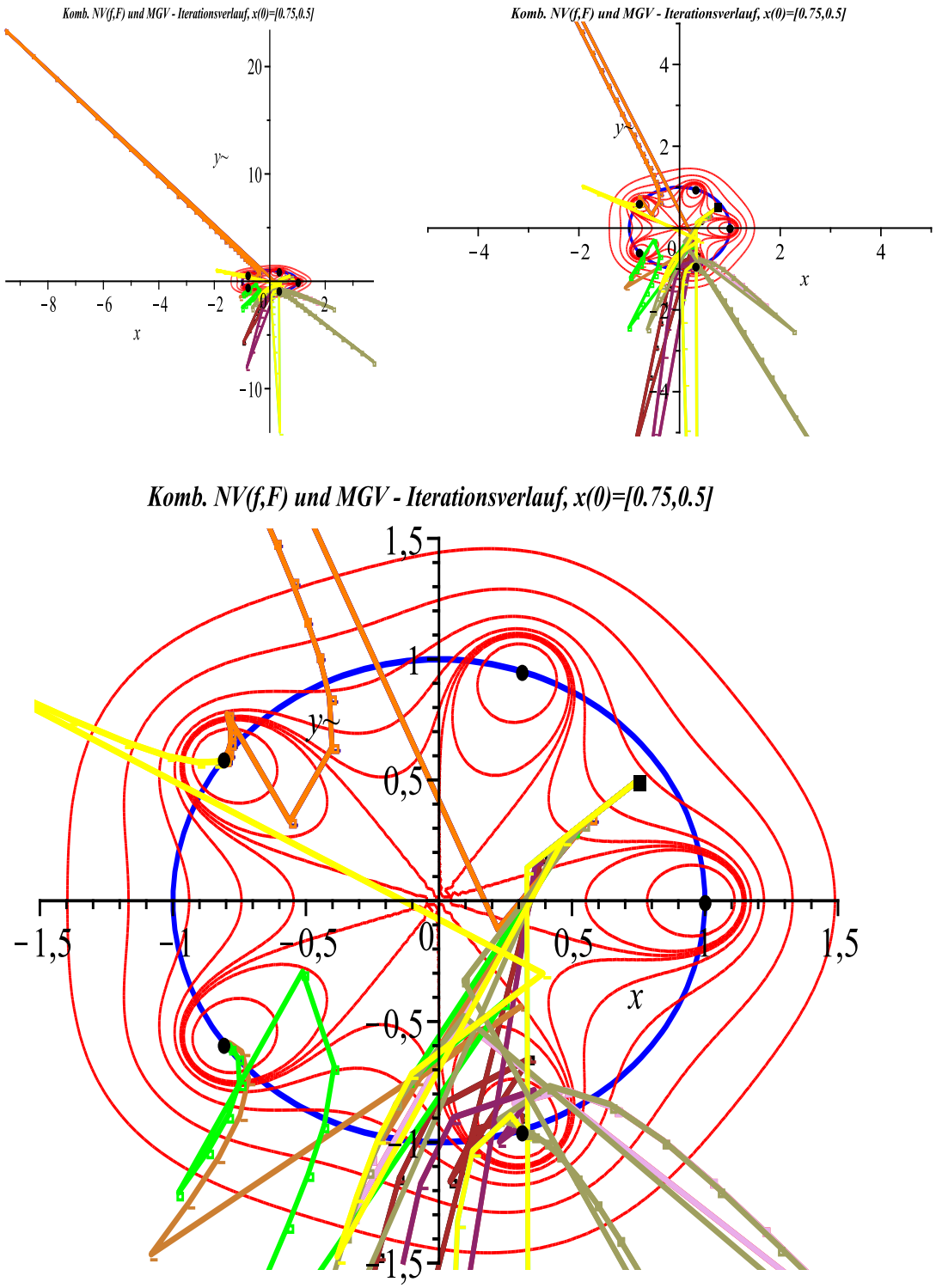


Abb. 3.25 Kombiniertes Lösungsverfahren mit $x^{(0)} = (0.5, 0.25)^T$, Iterationsverläufe, Einheitswurzeln, Höhenlinien `contours=[0.5,0.9,1,1.1,1.5,3,10,40]`, mit Ausschnittsvergrößerungen

Arbeitsblätter in Maple und MATLAB zu den Abschnitten

Abschnitt	Datei
1.2	Einheitswurzel0.mws
1.3	piecewise1.mws
1.4.1	koordinaten1.mws
1.4.2	koordinaten1.m polar2kart1.m kart2polar1.m kart2polar1a.m kart2polar2.m kart2polar2a.m
2	Newton_Verfahren1.mws
3.1	Einheitswurzel1.mws einzugsbereich_nv1.m
3.2	Einheitswurzel2.mws Einheitswurzel3.mws
3.3.1	kombn01.mws
3.3.2	kombn02.mws

Literaturverzeichnis

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

[2] Maess, G.: *Vorlesungen über Numerische Mathematik I, II*. Akademie-Verlag Berlin 1984, 1988.

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

[4] Abramowitz, M. und I. A. Stegun: *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications Inc., New York, 1992.

[5] Neundorf, W.: *Lösungsmethoden mit Maple. Betrachtung von Fehlern und Kondition sowie Darstellungsmöglichkeiten*. Preprint M 08/03 IfMath der TU Ilmenau, April 2003.

[6] Neundorf, W.: *Spezielle Aspekte zu CAS Maple und MATLAB. Rechengenauigkeit, Listen, Felder, Faktorisierungen, Dateiarbeit, TP → Maple, Maple → MATLAB mit Beispielen*. Preprint M 10/03 IfMath der TU Ilmenau, Juni 2003.

[7] Neundorf, W.: *Kondition eines Problems sowie Gleitpunktarithmetik in den CAS Maple, MATLAB und in höheren Programmiersprachen*. Preprint M 16/07 TUI, November 2007.

[8] Neundorf, W.: *MATLAB - Teil II: - Speicheraspekte, spezielle LGS, SDV, EWP, Graphik, NLG, NLGS*. Preprint M 23/99 TUI, September 1999.

[9] Neundorf, W.; Prätor, N.: *Modifiziertes Gradientenverfahren*. Preprint M 06/06 IfMath der TU Ilmenau, April 2006.

[10] Prätor, N.: *Kombinierte Lösungsverfahren für Gleichungssysteme*. Diplomarbeit IfMath der TU Ilmenau, Juni 2008.

[11] Vogt, W.: *Zur Numerik nichtlinearer Gleichungssysteme*. Teil 1: Preprint M 12/01 IfMath der TU Ilmenau, Oktober 2001; Teil 2: Preprint M 03/04 IfMath der TU Ilmenau, Februar 2004;

[12] B. Fine, B.; Rosenberger, G.: *The Fundamental Theorem of Algebra*. Springer 1997.

[13] Eisermann, M.: *The fundamental theorem of algebra made effective: an elementary real-algebraic proof via Sturm chains*. Preprint 2009, arXiv 0808.0097.
URL: www.igt.uni-stuttgart.de/eiserm

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