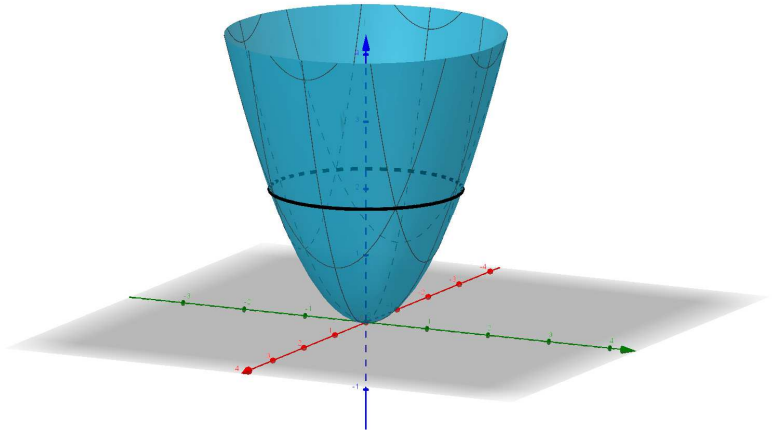
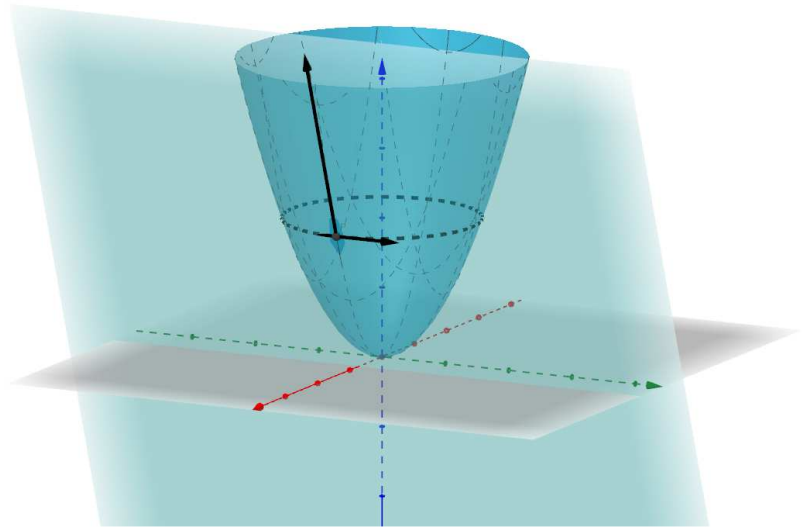
Geodätische von Funktionsgraphen

Ich orientiere mich am Buch von Christian Bär.[1] Als Beispiel soll uns der Graph $\Gamma \subseteq \mathbb{R}^3$ der Funktion $f(x,y) = x^2 + y^2$ dienen. Die Kurve $c(t) = \begin{pmatrix} \sqrt{2} \cdot cos(t) \\ \sqrt{2} \cdot sin(t) \\ 2 \end{pmatrix}$ beschreibt einen Kreis um die z-Achse in Höhe $z = 2$, der in Γ verläuft.



Weiter wählen wir den Punkt $P(\sqrt{2}|0|2)$ und die Tangentialebene E an den Graphen Γ in Punkt P. Sie wird aufgespannt von den Richtungsvektoren $t_x = \begin{pmatrix} 1 \\ 0 \\ \frac{\partial f}{\partial x} \end{pmatrix}$ und $t_y = \begin{pmatrix} 0 \\ 1 \\ \frac{\partial f}{\partial y} \end{pmatrix}$.



[1] C. Bär. *Elementare Differentialgeometrie*. De Gruyter Verlag, 2010.

Wenn wir eine an *P angeheftet gedachte* Linearkombinationen $v = \lambda \cdot t_x + \mu \cdot t_y$ beider Richtungsvektoren wählen und obige Prozedur für jeden Punkt der Kurve c wiederholen, dann erhalten wir ein Vektorfeld $v(t)$ mit der Eigenschaft, dass für jedes t_0 der Vektor $v(t_0)$ in der Tangentialebene an den Graphen Γ im Punkt $c(t_0)$ liegt. Ein für unsere Zwecke besonders wichtiges Vektorfeld mit dieser Eigenschaft ist das *Geschwindigkeitsfeld* $\dot{c}(t)$ unserer Kurve. Im Beispiel oben ist $\dot{c}(t) = \sqrt{2} \cdot t_y$.

In unserem Falle kann dann für ein differenzierbares Vektorfeld $v(t)$ die *kovariante Ableitung* definiert werden als eine *orthogonale Projektion* von $\dot{v}(t)$ auf die Tangentialebene E. Mit Hilfe des Skalarproduktes und einem Einheitsnormalenvektor n_E der Ebene lässt sie sich dann wie folgt schreiben.

$$\frac{\nabla}{dt} v(t) = \dot{v}(t) - \langle \dot{v}(t), n_E \rangle \cdot n_E$$

Wenn wir beispielsweise $v(t) = \dot{c}(t) = \begin{pmatrix} -\sqrt{2} \cdot sin(t) \\ \sqrt{2} \cdot cos(t) \\ 0 \end{pmatrix}$ setzen, dann erhalten wir für den Punkt *P* und seine Tangentialebene E den Einheitsnormalenvektor

$$n_E = \frac{t_x \times t_y}{\|t_x \times t_y\|} = \frac{1}{\|t_x \times t_y\|} \cdot \begin{pmatrix} -\frac{\partial f}{\partial x} \\ -\frac{\partial f}{\partial y} \\ 1 \end{pmatrix} = \frac{1}{\sqrt{8+0+1}} \cdot \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix} = \frac{1}{3} \cdot \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix}$$

sowie die Ableitung

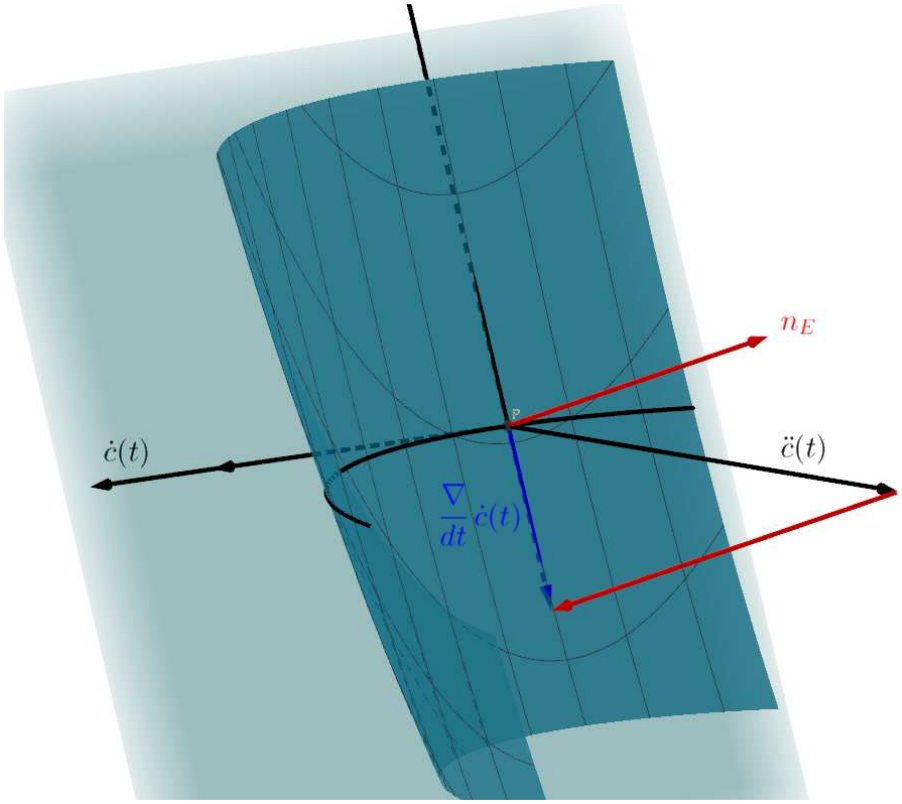
$$\dot{v}(t) = \ddot{c}(t) = \begin{pmatrix} -\sqrt{2} \cdot cos(t) \\ -\sqrt{2} \cdot sin(t) \\ 0 \end{pmatrix} \text{ bzw. } \dot{v}(0) = \begin{pmatrix} -\sqrt{2} \\ 0 \\ 0 \end{pmatrix} \text{ in Punkt } P.$$

Somit ist die kovariante Ableitung in *P* gegeben als

$$\frac{\nabla}{dt} \dot{c}(t) = \begin{pmatrix} -\sqrt{2} \\ 0 \\ 0 \end{pmatrix} - \frac{1}{9} \cdot \langle \begin{pmatrix} -\sqrt{2} \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix} \rangle \cdot \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix} =$$

$$= \begin{pmatrix} -\sqrt{2} \\ 0 \\ 0 \end{pmatrix} - \frac{4}{9} \cdot \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix} = -\frac{1}{9} \cdot \begin{pmatrix} \sqrt{2} \\ 0 \\ 4 \end{pmatrix}.$$

Nachfolgend ist ein Ausschnitt von Γ um den Punkt P herum mit den wichtigsten Akteuren gezeigt. Der eingezeichnete Vektor $\ddot{c}(t)$ zeigt lotrecht auf die z-Achse.



Für eine Geodätische muss die kovariante Ableitung des Geschwindigkeitsfeldes der Kurve verschwinden.

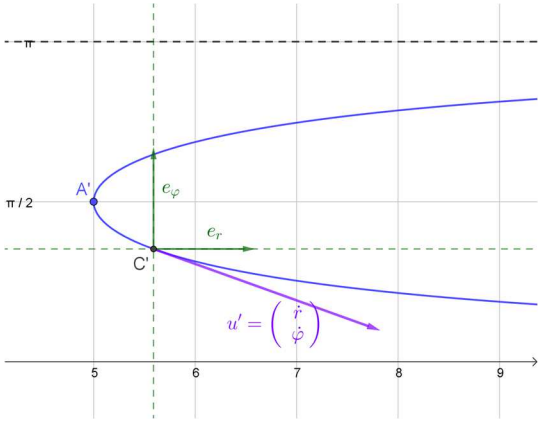
$$\frac{\nabla}{dt}\dot{c}(t) = 0$$

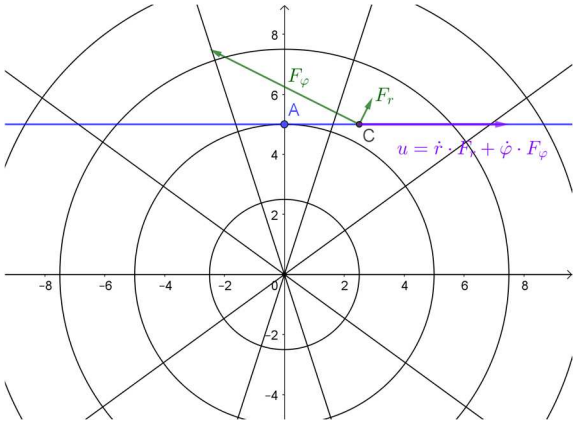
Damit wird erstens offenkundig, dass unsere Beispielkurve keine Geodätische ist. Zweitens eröffnen uns die angegebene Gleichung und die geometrische Interpretation der kovarianten Ableitung als orthogonale Projektion verhältnismäßig einfache Näherungslösungen der Geodätischen. Im Weiteren sei $v(t) = \dot{c}(t)$. Mit dem üblichen Differenzenquotienten schreibt sich die erste Ableitung von $v(t)$ wie folgt.

$$\dot{v}(t) = \lim_{h \to 0} \frac{v(t+h) - v(t)}{h}$$

Für genügend kleine h liest sich die Bedingung $0 = \frac{\nabla}{dt} v(t) = \dot{v}(t) - \langle \dot{v}(t), n_E \rangle \cdot n_E$ so, dass $\dot{v}(t)$ parallel zu n_E sein muss und daher (*in erster Näherung*) auch $v(t+h) - v(t)$. Dann aber können wir schreiben $v(t+h) = v(t) + \lambda \cdot n_E$ für ein noch aufzufindendes λ. Weiter verlangen wir von $v(t+h)$, dass er in der Tangentialebene am *neuen* Kurvenpunkt Q liegt. In einer früheren Version des Applets wurde Q ausgehend von einem Hilfspunkt $Q' = P + h \cdot v(t)$ ermittelt. Da Q' üblicherweise nicht mehr auf Γ liegt, fällte ich von ihm aus das Lot auf Γ und nahm den Schnitt des Lotes mit Γ als neuen Punkt Q. Mit der Bestimmung des Lotes gingen nicht nur erhebliche Komplikationen einher, ich hatte auch keine Rechtfertigung für diese Wahl des Punktes Q.

Der Graph Γ ist das Bild von $F: \mathbb{R}^2 \longrightarrow \mathbb{R}^3$, $P' = (x_P, y_P) \longmapsto \big(x_P, y_P, f(x_P, y_P)\big) = P$. Zu P' konstruieren wir nun eine Kurve $\gamma(\tau)$ mit $\gamma(0) = P'$ und $(F \circ \gamma)'(0) = v(t)$. Der Tangentenvektor an γ in P' soll also das Urbild von $v(t)$ in P sein. Für einen geeigneten Zeitschritt h' setzen wir schließlich $Q' = \gamma(h')$ und $Q = F(Q') \in \Gamma$. Nachfolgend ist die Situation gezeigt für die Polarkoordinatenabbildung $(r, \varphi) \longmapsto (r \cdot \cos(\varphi), r \cdot \sin(\varphi))$ und eine Gerade AC. Ihr Urbild ist die Kurve im linken Bild. Weiter sieht man den Zusammenhang der Tangentenvektoren u und u' an die Gerade beziehungsweise an ihr Urbild.





Für die Forderung $v(t) = (F \circ \gamma)'(0) = dF(P') \circ \gamma'(0)$ machen wir den Ansatz $\gamma'(\tau) = \begin{pmatrix} u(\tau) \\ v(\tau) \end{pmatrix}$ und setzen ein.

$$\begin{pmatrix} v^{(x)}(t) \\ v^{(y)}(t) \\ v^{(z)}(t) \end{pmatrix} = dF(P') \circ \gamma'(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ f_x(P') & f_y(P') \end{pmatrix} \circ \begin{pmatrix} u(0) \\ v(0) \end{pmatrix}$$

$$= \begin{pmatrix} u(0) \\ v(0) \\ f_x(P') \cdot u(0) + f_y(P') \cdot v(0) \end{pmatrix}$$

Hieraus lesen wir die Bedingungen $u(0) = v^{(x)}(t)$ und $v(0) = v^{(y)}(t)$ ab. Die einfachste Kurve, die unseren Ansprüchen genügt, ist daher $\gamma(\tau) = \begin{pmatrix} x_P \\ y_P \end{pmatrix} + \tau \cdot \begin{pmatrix} v^{(x)}(t) \\ v^{(y)}(t) \end{pmatrix}$. Mit ihr folgt

$$Q = F(Q') = F\big(\gamma(h')\big) = F\left(\left(x_P + h' \cdot v^{(x)}(t), y_P + h' \cdot v^{(y)}(t)\right)\right)$$

$$= \left(x_P + h' \cdot v^{(x)}(t), y_P + h' \cdot v^{(y)}(t), f\left(x_P + h' \cdot v^{(x)}(t), y_P + h' \cdot v^{(y)}(t)\right)\right)$$

$$\approx \Big(x_P + h' \cdot v^{(x)}(t), y_P + h' \cdot v^{(y)}(t), f(x_P, y_P) + f_x(P') \cdot h' \cdot v^{(x)}(t) + f_y(P') \cdot h' \cdot v^{(y)}(t)\Big)$$

$$= \Big(x_P + h' \cdot v^{(x)}(t), y_P + h' \cdot v^{(y)}(t), f(x_P, y_P) + h' \cdot v^{(z)}(t)\Big)$$

Die Differenz zu $P = \big(x_P, y_P, f(x_P, y_P)\big)$ ist also ungefähr gleich $h' \cdot v(t)$. Dies rechtfertigt die Wahl $h' = h$.

Jetzt können wir mit einem Normalenvektor n_F der Tangentialebene an Γ in Q die weitere Gleichung $\langle v(t+h), n_F\rangle = 0$ schreiben. Mit ihr erhalten wir $\langle v(t), n_F\rangle + \lambda \cdot \langle n_E, n_F\rangle = 0$, also die beiden Auflösungen

$$\lambda = -\frac{\langle v(t), n_F\rangle}{\langle n_E, n_F\rangle}$$

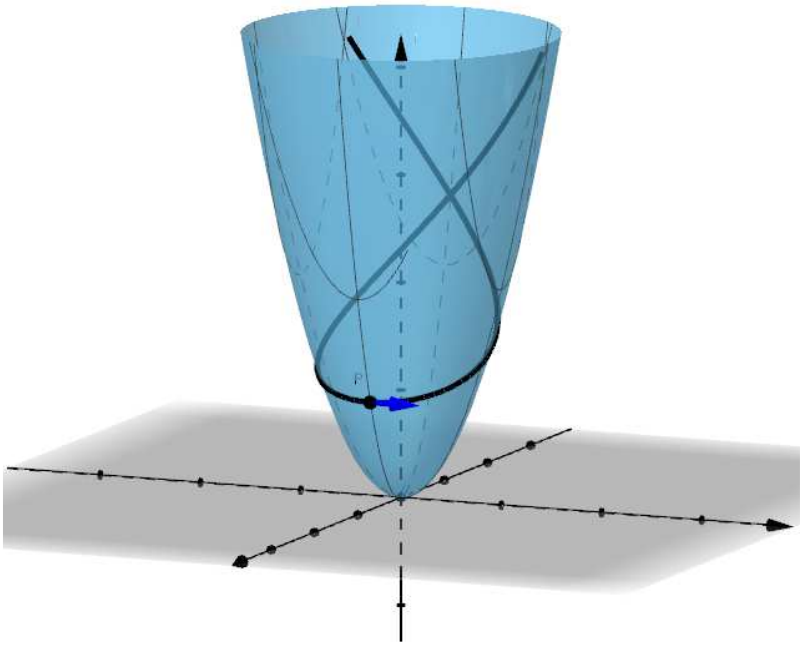
$$v(t+h) = v(t) - \frac{\langle v(t), n_F\rangle}{\langle n_E, n_F\rangle} \cdot n_E$$

Bemerkenswert für unsere anschließenden numerischen Berechnungen ist, dass wir für beide Normalenvektoren keine Einheitsvektoren benötigen, da Zähler und Nenner beide mit den Längen von n_E und n_F skalieren. Dafür werden in den Berechnungen die Tangentialvektoren auf Länge eins normiert, sodass h den ungefähren Abstand zweier Geodätenpunkte wiedergibt. Ein weiter offenes Problem ist die Verbindung der berechneten Punkte zu einer Kurve. Die einfachste (*und von mir bevorzugte*) Lösung ist die Verbindung zu einem Polygonzug im Raum um den Preis, dass die Kurve nicht im Graphen Γ verläuft.

Abschließend diskutieren wir unser Eingangsbeispiel für $h = 0{,}1$ und einhundert neu berechnete Punkte ($N = 100$) mit je fünfzig Zwischenpunkten ($n = 50$), die intern berechnet, aber nicht exportiert werden. Der letzte berechnete Punkt heiße R. Für seine Koordinaten und den zugehörigen Tangentialvektor entnimmt man *geogebra*

$$R(-2{,}076 \mid -1{,}948 \mid 8{,}104)$$

$$u = \begin{pmatrix} 0{,}232 \\ -0{,}466 \\ 0{,}855 \end{pmatrix}.$$



Da unser Graph rotationssymmetrisch ist, können wir den *Satz von Clairaut* auf ihn und seine Geodätische anwenden. Er besagt, dass das Produkt des Radius $r(t)$ des Breitenkreises durch einen Punkt $c(t)$ der Geodätischen und des Kosinus des von Geschwindigkeitsvektor $\dot{c}(t)$ und Breitenkreis eingeschlossenen Winkels $\vartheta(t)$ konstant ist.

$$r(t) \cdot cos\big(\vartheta(t)\big) = const.$$

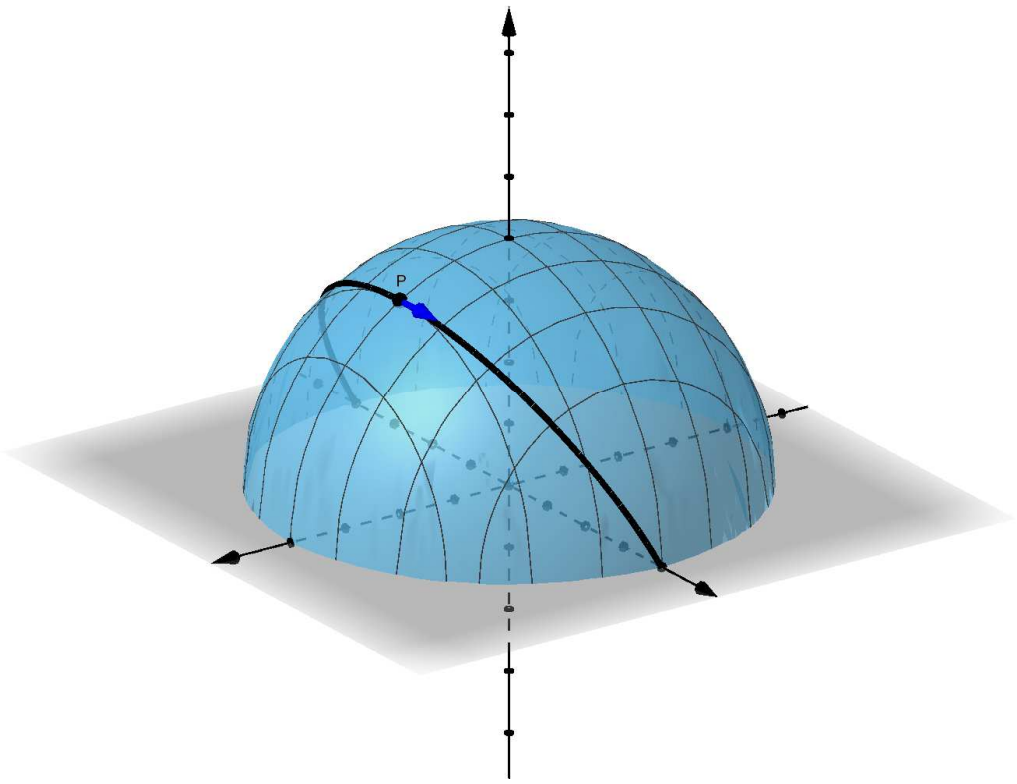
Für die Punkte P und R finden wir

$$r_P \cdot \langle v_P, t_P \rangle = \sqrt{2} \cdot \langle \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rangle = \sqrt{2} = 1{,}414 \ldots$$

$$r_R \cdot \langle v_R, t_R \rangle = \sqrt{(-2{,}076)^2 + (-1{,}948)^2} \cdot \langle \begin{pmatrix} 0{,}232 \\ -0{,}466 \\ 0{,}855 \end{pmatrix}, \begin{pmatrix} 0{,}684 \\ -0{,}729 \\ 0 \end{pmatrix} \rangle = 1{,}419$$

Damit ist der Wert für den Punkt R gerade einmal 0,3 % zu groß.

Ein weiteres schönes Beispiel stellt die Berechnung einer Geodätischen auf einer Kugel dar, die bekanntlich ein Großkreis ist.



Anhang

Javascript des Buttons *Berechnung der Geodätischen*

```
var i, j;          // Schleifenvariablen

var k;
var w;

var f_xn, f_yn, f_xn1, f_yn1; // Partielle Ableitungen von f an den Stellen (x_n, y_n) bzw. (x_(n+1), y_(n+1))

// Einlesen der benutzerdefinierten Werte

var x, y, z;              // x- und y-Koordinaten der Geodätenpunkte in beiden Richtungen
var x_P, y_P, z_P;     // x- und y-Koordinaten des Startpunktes P

var v_x, v_y, v_z;              // x- und y-Komponenten des Geschwindigkeitsvektors in beiden Richtungen
var v_xP, v_yP, v_zP;          // x- und y-Komponenten des Geschwindigkeitsvektors am Startpunkt P - für die "negative" Richtung ist eine Vorzeichenumkehr notwendig

x_P = ggbApplet.getValue( "x(P)" );          // x-Koordinate des Startpunktes P
y_P = ggbApplet.getValue(" y(P)" );
z_P = ggbApplet.getValue(" z(P)" );

v_xP = ggbApplet.getValue( "x(v)" );         // x-Komponente des Geschwindigkeitsvektors v
v_yP = ggbApplet.getValue(" y(v)" );
v_zP = ggbApplet.getValue(" z(v)" );

var h = ggbApplet.getValue(" h ");   // mittlerer Abstand zweier Geodätenpunkte
var N = ggbApplet.getValue(" N "); // Anzahl der (exportierten) Geodätenpunkte
var n = ggbApplet.getValue(" n ");   // Anzahl der (nicht exportierten) Zwischenpunkte zweier benachbarter Geodätenpunkte

h = h / n;                        // Rechnung mit "effektivem" h

// Setzen des Listenstartes auf den gemeinsamen Startpunkt P beider Geodätenabschnitte

ggbApplet.evalCommand( "M_p = { (" + x_P +  " , " + y_P + " , " + z_P + ") }" );
ggbApplet.evalCommand( "M_n = { (" + x_P +  " , " + y_P + " , " + z_P + " ) }" );
```

```
// Schleife mit Anzahl N Durchläufen zur Berechnung der Geodätenpunkte in Richtung von v

x = x_P;
y = y_P;

v_x = v_xP;
v_y = v_yP;
v_z = v_zP;

// In jedem Iterationsschritt werden die partiellen Ableitungen f_x und f_y an den Stellen
(x_n, y_n) bzw. (x_(n+1), y_(n+1)) berechnet. Im darauffolgenden Schritt nimmt die Stelle
(x_(n+1), y_(n+1)) die Rolle von (x_n, y_n) ein. Um unnötigen Rechenaufwand zu vermei-
den, werden f_xn und f_yn mit den Werten von f_xn1 bzw. f_yn1 überschrieben. Hierzu be-
darf es aber initialer Werte...
f_xn1 = ggbApplet.getValue(" f_x(  " + x + " ,  " + y + " ) ");
f_yn1 = ggbApplet.getValue(" f_y(  " + x + " ,  " + y + " ) ");

for( j = 0; j < N; j++ )
{
        // Schleife mit n Iterationsschritten

        for( i = 0; i < n; i++)
        {
                // Länge des Geschwindigkeitsvektors
                w = Math.sqrt( v_x*v_x + v_y*v_y + v_z*v_z );

                // Berechnung der partiellen Ableitungen von f am aktuellen Geodätenpunkt
                f_xn = f_xn1;
                f_yn = f_yn1;

                // Berechnung des neuen Geodätenpunktes - ohne z-Koordinate
                x = x + h*v_x / w;
                y = y + h*v_y / w;

                // Berechnung der partiellen Ableitungen von f am neuen Geodätenpunkt
                f_xn1 = ggbApplet.getValue(" f_x(  " + x + " ,  " + y + " ) ");
                f_yn1 = ggbApplet.getValue(" f_y(  " + x + " ,  " + y + " ) ");

                // Nebenrechnung für Vorfaktor k in der Berechnung des neuen Geschwindig-
keitsvektors
                k = ( v_x *( f_xn - f_xn1 ) + v_y *( f_yn - f_yn1 ) ) / ( 1 + f_xn*f_xn1 +
f_yn*f_yn1 );

                // Berechnung des neuen Geschwindigkeitsvektors
                v_x = v_x + k*f_xn;
                v_y = v_y + k*f_yn;
                v_z = v_z - k;
}       // Ende der Iterationsschleife
```

```
        // Export des aktuellen Geodätenpunktes - Anhängen an die Liste M_p

        z = ggbApplet.getValue(" f(  " + x + " ,  " + y + " ) ");

        ggbApplet.evalCommand("L = CopyFreeObject(Append(M_p, (  " + x + " ,  " + y + ",
" + z + " )))");
        ggbApplet.deleteObject("M_p");
        ggbApplet.evalCommand("Rename(L, M_p)");

}       // Ende der Schleife

// Schleife mit Anzahl N Durchläufen zur Berechnung der Geodätenpunkte gegen die Rich-
tung von v

x = x_P;
y = y_P;

v_x = -v_xP;
v_y = -v_yP;
v_z = -v_zP;

// Wie oben erläutert, werden f_xn1 und f_yn1 initiale Werte zum Überschreiben von f_xn
und f_yn zugeordnet...
f_xn1 = ggbApplet.getValue(" f_x(  " + x + " ,  " + y + " ) ");
f_yn1 = ggbApplet.getValue(" f_y(  " + x + " ,  " + y + " ) ");

for( j = 0; j < N; j++ )
{
        // Schleife mit n Iterationsschritten

        for( i = 0; i < n; i++)
        {
                // Länge des Geschwindigkeitsvektors
                w = Math.sqrt( v_x*v_x + v_y*v_y + v_z*v_z );

                // Berechnung der partiellen Ableitungen von f am aktuellen Geodätenpunkt
                f_xn = f_xn1;
                f_yn = f_yn1;

                // Berechnung des neuen Geodätenpunktes - ohne z-Koordinate
                x = x + h*v_x / w;
                y = y + h*v_y / w;

                // Berechnung der partiellen Ableitungen von f am neuen Geodätenpunkt
                f_xn1 = ggbApplet.getValue(" f_x(  " + x + " ,  " + y + " ) ");
                f_yn1 = ggbApplet.getValue(" f_y(  " + x + " ,  " + y + " ) ");
```

```
                // Nebenrechnung für Vorfaktor k in der Berechnung des neuen Geschwindig-
keitsvektors
                k = ( v_x *( f_xn - f_xn1 ) + v_y *( f_yn - f_yn1 ) ) / ( 1 + f_xn*f_xn1 +
f_yn*f_yn1 );

                // Berechnung des neuen Geschwindigkeitsvektors
                v_x = v_x + k*f_xn;
                v_y = v_y + k*f_yn;
                v_z = v_z - k;

        }       // Ende der Iterationsschleife

        // Export des aktuellen Geodätenpunktes - Anhängen an die Liste M_p

        z = ggbApplet.getValue(" f(  " + x + " ,  " + y + " ) ");

        ggbApplet.evalCommand("L = CopyFreeObject(Append(M_n, (  " + x + " ,  " + y + ",
" + z + " )))");
        ggbApplet.deleteObject("M_n");
        ggbApplet.evalCommand("Rename(L, M_n)");

}       // Ende der Schleife

// Verbinden der berechneten Punkte zu Polygonzügen

ggbApplet.evalCommand( " G_p = Polyline(M_p) " );
ggbApplet.setLabelVisible( "G_p", false );
ggbApplet.evalCommand( " G_n = Polyline(M_n) " );
ggbApplet.setLabelVisible( "G_n", false );
```