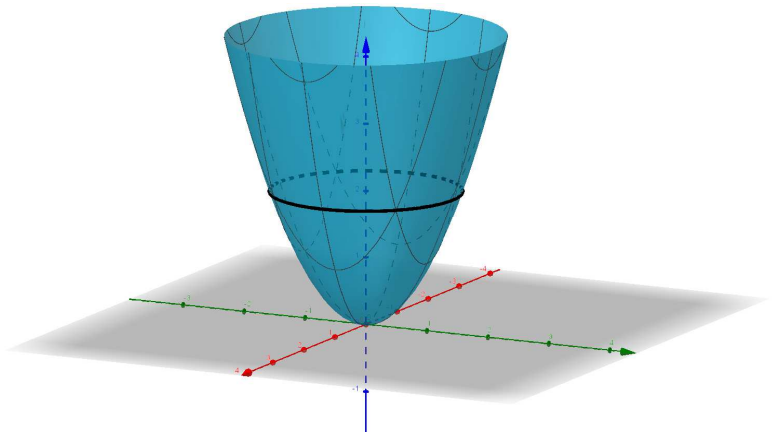
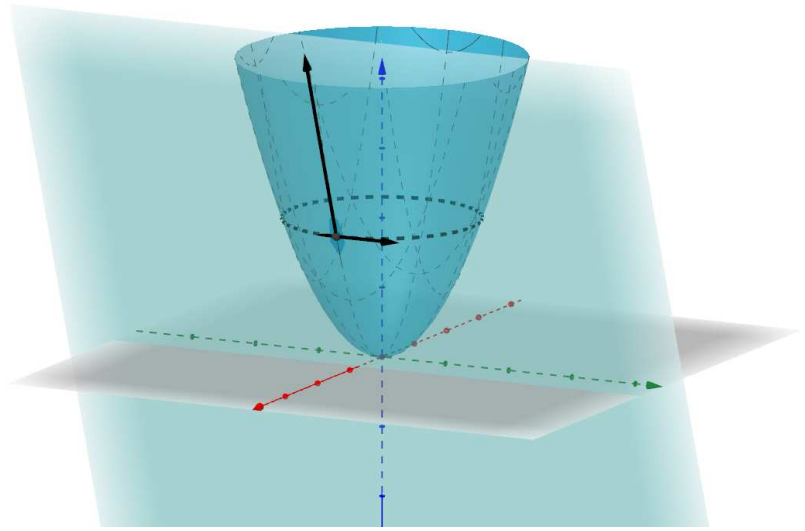
Geodätische von Funktionsgraphen

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

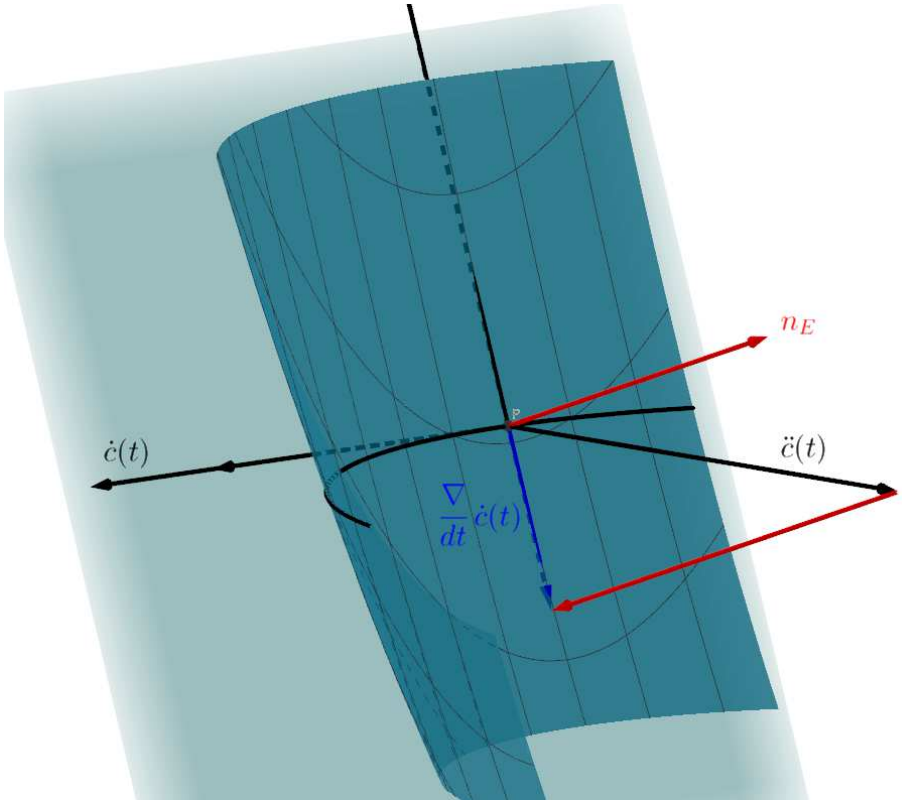
$$\dot{v}(t) = \ddot{c}(t) = \begin{pmatrix} -\sqrt{2} \cdot cos(t) \\ -\sqrt{2} \cdot sin(t) \\ 0 \end{pmatrix} \text{ bzw. } \dot{v}(0) = \begin{pmatrix} -\sqrt{2} \\ 0 \\ 0 \end{pmatrix} \text{ in Punkt } P.$$

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.

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

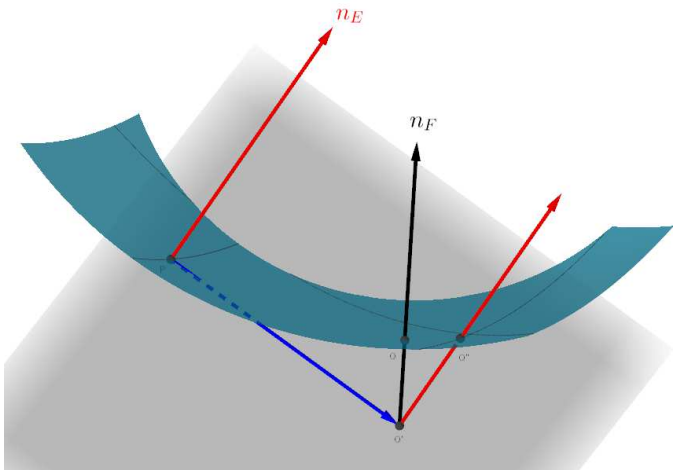
Für genügend kleine h liest sich die Bedingung $0 = \frac{\nabla}{dt} v(t) = \dot{v}(t) - \langle \dot{v}(t), n_E \rangle \cdot n_E$ so, dass $\dot{v}(t)$ parallel zu n_E sein muss und daher (*in erster Näherung*) auch $v(t+h) - v(t)$. Dann aber können wir schreiben $v(t+h) = v(t) + \lambda \cdot n_E$ für ein noch aufzufindendes λ. Weiter verlangen wir von $v(t+h)$, dass er in der Tangentialebene am *neuen* Kurvenpunkt Q liegt. Letzteren ermitteln wir ausgehend vom Hilfspunkt $Q' = P + h \cdot v(t)$. Da Q' üblicherweise nicht mehr auf Γ liegt, fällen wir von ihm aus das Lot auf Γ und nehmen den Schnitt des Lotes mit Γ als neuen Punkt Q. Leider bietet *geogebra* hierfür kein geeignetes Werkzeug an und wir müssen improvisieren.

- Wir folgen einer schönen Idee zur Auffindung von Extrema in Analogie zum Newtonverfahren[2] und minimieren den Abstand des Punktes Q' zu Γ. Hierzu suchen wir ein Minimum der Funktion

$$d(x,y) = \left(x - x_{Q'}\right)^2 + \left(y - y_{Q'}\right)^2 + \left(f(x,y) - z_{Q'}\right)^2$$

- Als Startpunkt wählen wir den Schnitt der Geraden $\{Q' + \lambda \cdot n_E \mid \lambda \in \mathbb{R}\}$ mit Γ, worin n_E der *alte* Normalenvektor ist. Zur Auffindung des Schnittpunktes lassen wir *geogebra* denjenigen Wert von λ suchen, für welchen die Differenz der z-Koordinaten von Geraden- und Graphenpunkten (*zu den gleichen x- und y-Koordinaten*) verschwindet, also die Nullstelle der Funktion $f\left(x_{Q'} + \lambda \cdot n_E^{(x)}, y_{Q'} + \lambda \cdot n_E^{(y)}\right) - \left(z_{Q'} + \lambda \cdot n_E^{(z)}\right)$ mit Startwert $\lambda = 0$.

- Die Prozedur der Minimierung ist in Anhang A näher ausgeführt und liefert (*näherungsweise*) den Punkt Q.



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

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

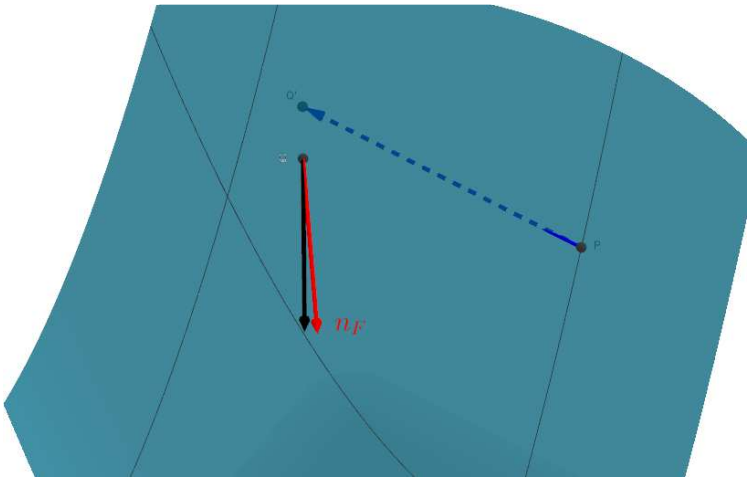
$$v(t+h) = v(t) - \frac{\langle v(t), n_F \rangle}{\langle n_E, n_F \rangle} \cdot n_E$$

Bemerkenswert für unsere anschließenden numerischen Berechnungen ist, dass wir für beide Normalenvektoren keine Einheitsvektoren benötigen. Im letzten Bild wurde $h = 1$ gewählt. Für den Startpunkt Q'' der Iteration findet *geogebra* die Koordinaten (1,061 | 1 | 2,125). Hieraus berechnete es (*in zehn Schritten*) die Näherung Q(1,178 | 0,845 | 2,101). Die Güte dieser Näherung lässt sich durch den Vergleich der beiden (*auf Länge eins normierten*) Vektoren n_F und $\overrightarrow{Q'Q}$ abschätzen.

[2] https://www.geogebra.org/m/efnxsX4Z

$$\overrightarrow{Q'Q^0} = \begin{pmatrix} -0{,}788 \\ -0{,}517 \\ 0{,}335 \end{pmatrix}$$

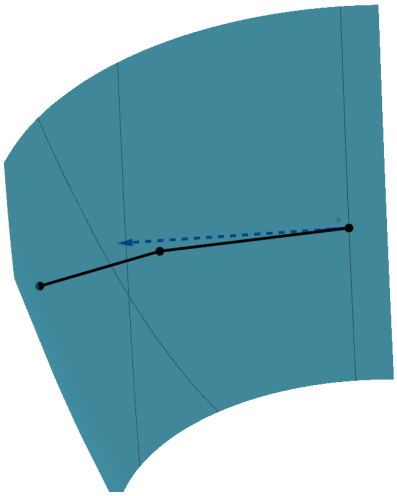
$$n_F = \begin{pmatrix} -0{,}768 \\ -0{,}551 \\ 0{,}326 \end{pmatrix}$$



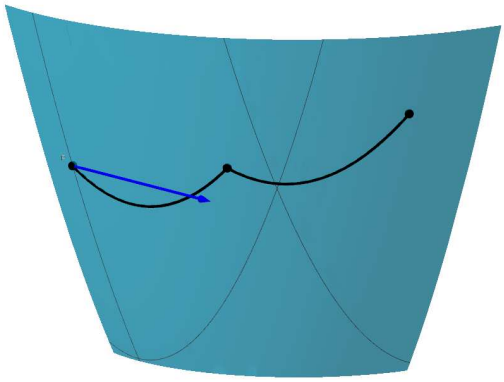
Abschließend können wir den neuen Tangentialvektor an die Geodätische in Q berechnen.

$$\begin{aligned} v(1) = \; & v(0) - \frac{\langle v(0), n_F \rangle}{\langle n_E, n_F \rangle} \cdot n_E \\ & = \begin{pmatrix} 0 \\ \sqrt{2} \\ 0 \end{pmatrix} - \frac{\sqrt{2} \cdot (-0{,}551)}{\frac{1}{3} \cdot \left(-2\sqrt{2} \cdot (-0{,}768) + 1 \cdot 0{,}326\right)} \cdot \frac{1}{3} \cdot \begin{pmatrix} -2\sqrt{2} \\ 0 \\ 1 \end{pmatrix} \\ & = \begin{pmatrix} -0{,}882 \\ 1{,}414 \\ 0{,}312 \end{pmatrix} \end{aligned}$$

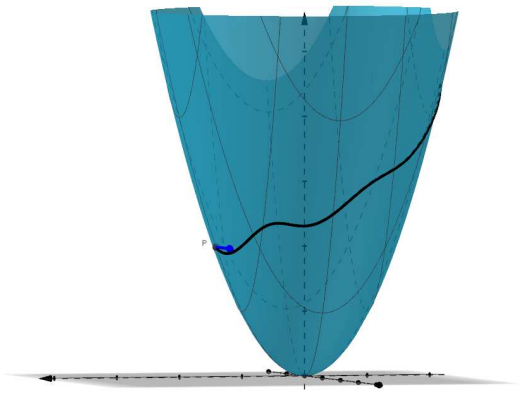
In den Berechnungen werden die Tangentialvektoren noch auf Länge eins normiert. Ein weiter offenes Problem ist die Verbindung der berechneten Punkte zu einer Kurve. Die einfachste (*und von mir bevorzugte*) Lösung ist die Verbindung zu einem Polygonzug im Raum um den Preis, dass die Kurve nicht im Graphen Γ verläuft.



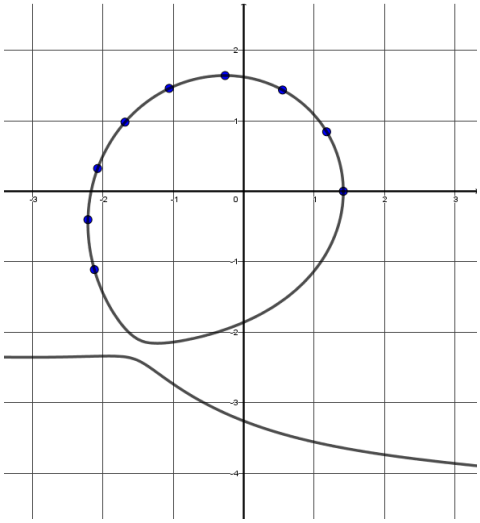
Werden die Punkte erst in die xy-Ebene projiziert, dort zu einem Polygonzug verbunden und dann zu diesem stückweise eine Kurve in Γ errechnet, so ergibt sich eine Girlande.



Ein mit dem Befehl *TrendPoly*(<*Liste von Punkten*>, <*Grad des Polynoms*>) erzeugter Graph zu den in die xy-Ebene projizierten Punkten liefert gleichfalls kein zufriedenstellendes Ergebnis – neben weiteren Problemen.



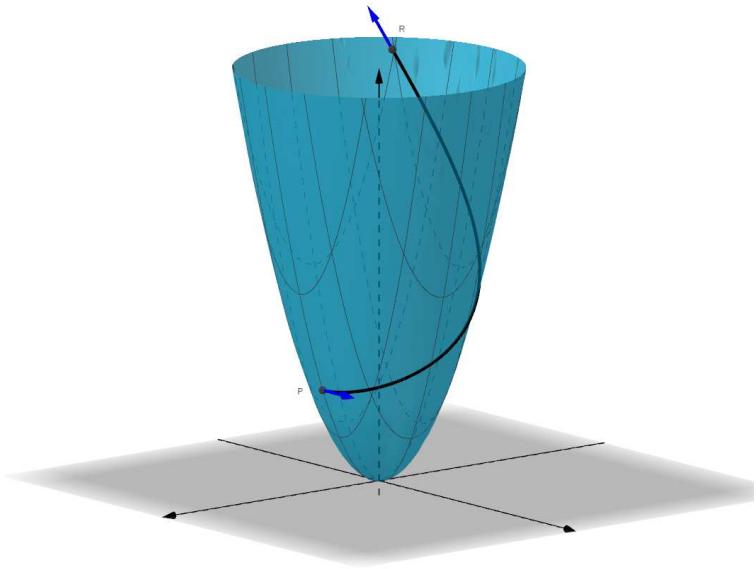
Die erfreulichsten Ergebnisse lieferte der Befehl *ImpliziteKurve*(<*Liste von Punkten*>); zu beachten ist die Einschränkung auf Listen mit $\frac{n\cdot(n+3)}{2}$ Punkten. Bedauerlicherweise kenne ich keinen Zugriff auf die Punkte der so erzeugten Kurve.



Abschließend diskutieren wir unser Eingangsbeispiel für $h = 0{,}1$ und einhundert neu berechnete Punkte ($N = 100$) mit je fünf Iterationsschritten ($n = 5$) für die Auffindung der Lotfußpunkte. Der letzte berechnete Punkt heiße R. Für seine Koordinaten und den zugehörigen Tangentialvektor entnimmt man *geogebra*

$$R(\,-1{,}976 \mid -2{,}029 \mid 8{,}020\,)$$

$$u = \begin{pmatrix} 0{,}261 \\ -0{,}463 \\ 0{,}847 \end{pmatrix}.$$



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.

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

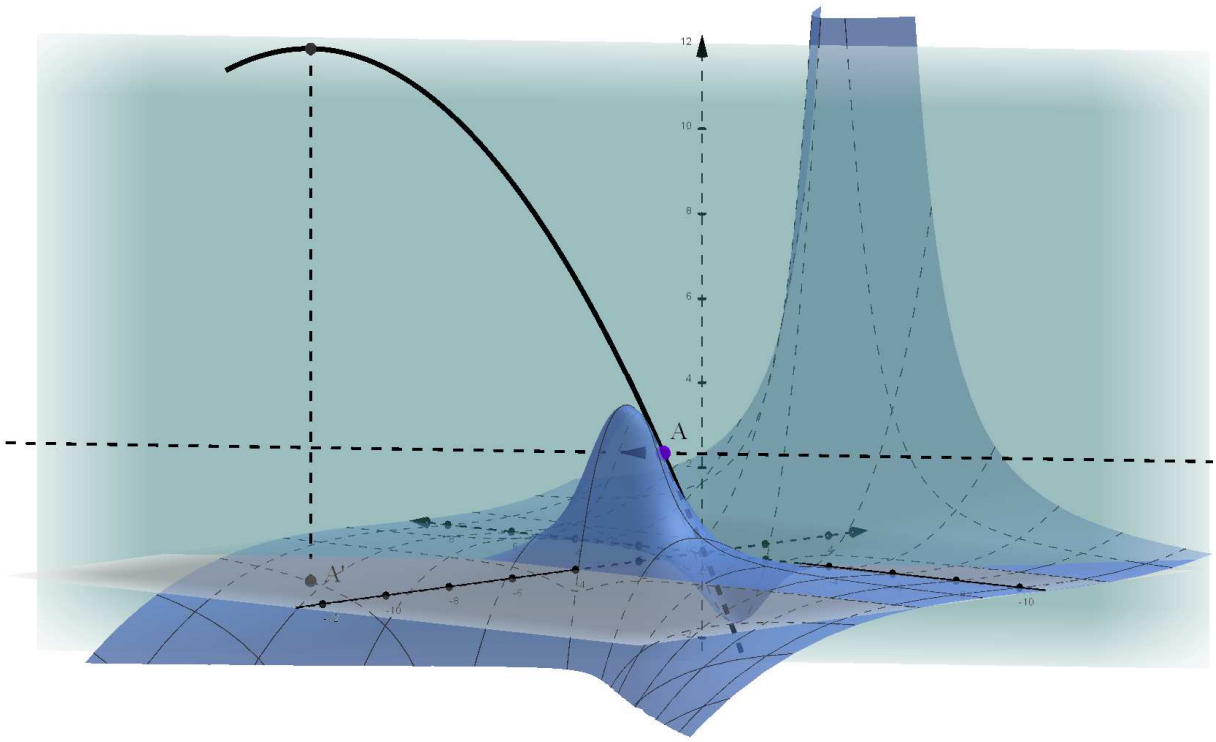
Für die Punkte P und R finden wir

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

$$r_R \cdot \langle v_R, t_R \rangle = \sqrt{(-1{,}976)^2 + (-2{,}029)^2} \cdot \langle \begin{pmatrix} 0{,}261 \\ -0{,}463 \\ 0{,}847 \end{pmatrix}, \begin{pmatrix} 0{,}716 \\ -0{,}698 \\ 0 \end{pmatrix} \rangle = 1{,}445$$

Damit ist der Wert für den Punkt R gerade einmal 2 % zu groß.

Bei allen Erfolgen fehlen noch etliche Verfeinerungen. Zum Beispiel kann ich die Zuverlässigkeit der Minimumsuche nach dem in Anhang A beschriebenen Verfahren leider nicht abschätzen. Das folgende Bild zeigt den Graphen der Funktion $f(x,y)=\frac{x^3+2x^2-8x-2}{4y^2+4}$ und den Startpunkt $A(-1{,}71|-0{,}47)$. Offenkundig verfehlt die Näherungsprozedur den Hochpunkt weit. Der Punkt A' liegt ferner von Punkt A als der benachbarte Tiefpunkt.



Ich freue mich schon auf neue *geogebra-Aktivitäten* zu Extrema und Geodätischen!

Anhang A

Gesucht werde ein Extremum der Funktion $f(x,y)$ ausgehend von einem Startpunkt $(x_0|y_0)$. Da der Gradient von f in Richtung der größten Änderungen zeigt, werde der Definitionsbereich von f auf die Gerade $\left\{\left(x_0 + t \cdot f_x(x_0,y_0)\middle| y_0 + t \cdot f_y(x_0,y_0)\right) \mid t \in \mathbb{R}\right\}$ beschränkt. Hierdurch ist eine neue Funktion *s* bestimmt mit

$$s(t) = f\left(x_0 + t \cdot f_x(x_0,y_0), y_0 + t \cdot f_y(x_0,y_0)\right)$$

Zu ihr wird für $t = 0$ eine Schmiegparabel mit der Gleichung $g(t) = pt^2 + qt + r$ gesucht. Dies führt auf die drei Gleichungen

$$s(0) = g(0) \quad \Leftrightarrow \quad f(x_0,y_0) = r$$

$$s'(0) = g'(0) \quad \Leftrightarrow \quad df(x_0,y_0) \circ \begin{pmatrix} f_x(x_0,y_0) \\ f_y(x_0,y_0) \end{pmatrix} = q$$

$$\Leftrightarrow \quad \left(f_x(x_0,y_0)\right)^2 + \left(f_y(x_0,y_0)\right)^2 = q$$

$$s''(0) = g''(0) \quad \Leftrightarrow \quad s''(0) = 2p$$

Zum Ausschreiben der letzten Gleichung werden die Abkürzungen $\varphi_x = f_x(x_0,y_0)$ und $\varphi_y = f_y(x_0,y_0)$ eingeführt. Es ist dann

$$s'(t) = \varphi_x \cdot f_x\left(x_0 + t \cdot \varphi_x, y_0 + t \cdot \varphi_y\right) + \varphi_y \cdot f_y\left(x_0 + t \cdot \varphi_x, y_0 + t \cdot \varphi_y\right) \text{ und}$$

$$s''(t) = \varphi_x \cdot df_x \circ \begin{pmatrix} \varphi_x \\ \varphi_y \end{pmatrix} + \varphi_y \cdot df_y \circ \begin{pmatrix} \varphi_x \\ \varphi_y \end{pmatrix}.$$

Auswertung der letzten Zeile an der Stelle $t = 0$ liefert

$$s''(0) = {\varphi_x}^2 \cdot f_{xx}(x_0,y_0) + 2\varphi_x\varphi_y \cdot f_{xy}(x_0,y_0) + {\varphi_y}^2 \cdot f_{yy}(x_0,y_0) = 2p.$$

Die Scheitelkoordinate $t_S = -\frac{q}{2p}$ liefert abschließend die neue (*und hoffentlich bessere*) Näherung $(x_1|y_1)$ mit $x_1 = x_0 + t_S \cdot f_x(x_0,y_0)$ und $y_1 = y_0 + t_S \cdot f_y(x_0,y_0)$.

Anhang B

Javascript des Buttons *Berechnung der Geodätischen*

```
// Benötigte Variablen für Berechnungen

var x_E, y_E, z_E;     // Punkt E
var x_P, y_P, z_P;     // Punkt P

var x_u, y_u, z_u;     // Neuer Tangentialvektor
var x_g, y_g;          // Koordinaten des Gradienten in E
var n_x, n_y, n_z;     // Normalenvektor auf Tangentialebene in D
var m_x, m_y, m_z;     // Normalenvektor auf Tangentialebene in E

var l_p;               // erste Schätzung des Abstandes der Spitze des Tangentialvektors zum
                       Graphen
var k;          // Benötigtes Lambda für Projektion des neuen Tangentialvektors in die alte
                Tangentialebene
var t;          // Vielfaches des Einheitsgradienten zur Näherung des neuen Punktes
var w;          // Länge des Tangentialvektors v

var a;          // Wert der Funktion am Punkt E
var q_x, q_y, q_xx, q_yy, q_xy;      // Werte der partiellen Ableitungen am Punkt E einer
                                        speziellen Abstandsfunktion in Iterationsschleife
var p, q;                            // Koeffizienten des quadratischen Näherungspolynoms

// Einlesen der benutzerdefinierten Werte

var x_D = ggbApplet.getValue(" x(D) ");    // x-Koordinate des Startpunktes D
var y_D = ggbApplet.getValue(" y(D) ");
var z_D = ggbApplet.getValue(" z(D) ");

var x_v = ggbApplet.getValue(" x(v) ");     // x-Koordinate des Tangentenvektors in D
var y_v = ggbApplet.getValue(" y(v) ");
var z_v = ggbApplet.getValue(" z(v) ");

var h = ggbApplet.getValue(" h ");   // mittlerer Abstand zweier Geodätenpunkte
var N = ggbApplet.getValue(" N "); // Anzahl der Geodätenpunkte
var n = ggbApplet.getValue(" n ");   // Anzahl der Schleifendurchgänge für Näherungen des
                                        neuen Geodätenpunktes

// Löschen der alten Geodäten und Setzen des Listenstartes auf den Startpunkt der Geodäten D

ggbApplet.deleteObject("M");
ggbApplet.evalCommand( "M = { (" + x_D +  " , " + y_D + " , " + z_D + " ) }" );
```

```
// Schleife mit Anzahl N Durchläufen zur Berechnung der Geodätenpunkte

for( var i = 0; i < N; i++)
{

// Normalisierung des Tangentialvektors

w = Math.sqrt(x_v*x_v + y_v*y_v + z_v*z_v);
x_v = x_v / w;
y_v = y_v / w;
z_v = z_v / w;

// Berechnung des Normalenvektors an die Tangentialebene in D

ggbApplet.evalCommand("b_{xy}=b(" + x_D + "," + y_D + ")");
n_x = (-1) * ggbApplet.getValue("b_{xy}");
ggbApplet.evalCommand("c_{xy}=c(" + x_D + "," + y_D + ")");
n_y = (-1) * ggbApplet.getValue("c_{xy}");
n_z = 1;

// Normalisierung des Normalenvektors in D

w = Math.sqrt(n_x*n_x + n_y*n_y + n_z*n_z);
n_x = n_x / w;
n_y = n_y / w;
n_z = n_z / w;

// Berechnung des Punktes P an der Spitze des Tangentialvektors

x_P = x_D + h*x_v;
y_P = y_D + h*y_v;
z_P = z_D + h*z_v;

// In erster Näherung des Lotes von P auf den Graphen wird für den Richtungsvektor der alte
Lotvektor in D angenommen.
// Erzeugung der Funktion p(x) des z-Abstandes der Punkte auf der Näherungsgeraden zum
Graphen und Berechnung der zu null nächsten Nullstelle

ggbApplet.evalCommand("p(x) =  a( " + x_P + " + x*" + n_x + " , " + y_P + " + x*" + n_y +"
) - " + z_P + " - x*" + n_z + "  ");
ggbApplet.evalCommand("l_p = x( Root(p,0))");
l_p = ggbApplet.getValue("l_p");

// Berechnung des neuen Punktes E auf der Geodäten - erste Schätzung mit l_p

x_E = x_P + l_p*n_x;
y_E = y_P + l_p*n_y;
```

```
// ITERATIONSSCHLEIFE FÜR EINE PRÄZISERE NÄHERUNG

// Erzeugung der Funktion q(x) des Abstandes von P zum Graphen

ggbApplet.evalCommand( " q(x,y) = (x - " + x_P + ")^2 + (y - " + y_P + ")^2 + ( a( x , y ) - "
+ z_P +  " )^2 " );

        // Schleife mit n Iterationsschritten

        for( var k = 0; k < n; k++)
        {

        // Einlesen der zum Punkt E gehörigen Werte von Gradient und Funkitonen

        ggbApplet.evalCommand("q_{0x} = q_x(" + x_E + "," + y_E + ")");
        q_x =  ggbApplet.getValue("q_{0x}");
        ggbApplet.evalCommand("q_{0y} = q_y(" + x_E + "," + y_E + ")");
        q_y =  ggbApplet.getValue("q_{0y}");
        ggbApplet.evalCommand("q_{0xx} = q_{xx}(" + x_E + "," + y_E + ")");
        q_xx = ggbApplet.getValue("q_{0xx}");
        ggbApplet.evalCommand("q_{0yy} = q_{yy}(" + x_E + "," + y_E + ")");
        q_yy = ggbApplet.getValue("q_{0yy}");
        ggbApplet.evalCommand("q_{0xy} = q_{xy}(" + x_E + "," + y_E + ")");
        q_xy = ggbApplet.getValue("q_{0xy}");

        // Normalisierung des Gradienten

        w = Math.sqrt(q_x*q_x + q_y*q_y);
        x_g = q_x / w;
        y_g = q_y / w;

        // Berechnung der Koeffizienten des quadratischen Näherungspolynoms

        q = w;
        p = 0.5 * ( q_xx*q_x*q_x + 2*q_xy*q_x*q_y + q_yy*q_y*q_y ) / ( w*w );

        // Berechnung des neuen Punktes E

        t = (-1)*q / 2 / p;
        x_E = x_E + t*x_g;
        y_E = y_E + t*y_g;

        }       // Ende der Iterationsschleife

// Berechnung der z-Koordinate der letzten neuen Näherung von E
```

```
ggbApplet.evalCommand("a_0 = a(" + x_E + "," + y_E + ")");
z_E =  ggbApplet.getValue("a_0");

// ENDE DER ITERATIONSROUTINE



// Export des Punktes E - Anhängen an die Liste M

ggbApplet.evalCommand("L = CopyFreeObject(Append(M, (  " + x_E + " , " + y_E + "  ,  " +
z_E + " )))");
ggbApplet.deleteObject("M");
ggbApplet.evalCommand("Rename(L, M)");

// Berechnung des Normalenvektors an die Tangentialebene in E

ggbApplet.evalCommand("b_{xy}=b(" + x_E + "," + y_E + ")");
m_x = (-1) * ggbApplet.getValue("b_{xy}");
ggbApplet.evalCommand("c_{xy}=c(" + x_E + "," + y_E + ")");
m_y = (-1) * ggbApplet.getValue("c_{xy}");
m_z = 1;

// Berechnung von Lambda

k = (x_v*m_x + y_v*m_y + z_v*m_z) / (n_x*m_x + n_y*m_y + n_z*m_z);

// Berechnung des neuen Tangentialvektors

x_u = x_v - k*n_x;
y_u = y_v - k*n_y;
z_u = z_v - k*n_z;

// Überschreiben der alten Werte von D und v mit den neuen Werten von E und u

x_D = x_E;
y_D = y_E;
z_D = z_E;

x_v = x_u;
y_v = y_u;
z_v = z_u;

}         // Ende der Schleife

// Verbinden der berechneten Punkte zu einem Polygonzug

ggbApplet.evalCommand( " G = Polyline(M) " );
```