Computerpraktikum

Ebene Kurven

Bei -

Von -

Mit -

Wintersemester 2022/2023

30.11.2022

Inhaltsverzeichnis

0 Vorwort

Dieser Teil ist eine Ansammlung von mehr oder weniger unzusammenhängenden Anmerkungen und Gedanken zu dem Projekt und kann bis auf den folgenden Teil 0.1 Notation übersprungen werden.

Mein Partner Manuel Müller und ich haben uns mit Projekt 1 „Ebene Kurven und Kurvenscharen“[1] beschäftigt.

Für das Projekt wurde *Maple 2022* verwendet. Ich weiß leider nicht, wie es mit der Rückwärtskompatibilität aussieht. Falls Sie Probleme mit einem der Skripte haben, können Sie sich gerne an mich wenden, ansonsten liegen diesem Dokument einige *Maple*-Worksheets bei, welche sich mit den einzelnen Teilaufgaben befassen[2].
Dieses Dokument wurde in LaTeX verfasst, da man hier meiner Meinung nach viel schönere Texte, als in *Maple* hinbekommt und die LaTeX-Umgebung in *Maple* einfach nicht so schön ist.
Im Text wird manchmal auf Fußnoten[3] verwiesen. Dies sind absolut unnötige und unformale Kommentare von mir. Sie können Sie gerne lesen, meistens sind sie erheiternd. Finden können Sie diese ganz am Ende auf der letzten Seite.

Ich bitte Sie darum gerne mit den Programmen rumzuspielen und eigene Kurven und Funktionen auszuprobieren. Wir haben uns bemüht für jeden Teil der Aufgabenstellung einen schönen Plot oder eine wunderbare Animation zu machen.
Mit den von uns gewählten Beispielen sollte es auch problemlos möglich sein den Syntax der Prozedur zu verstehen. Die meisten Funktionen benötigen eine Kurve $\mathtt{c}(\mathtt{t}) :=< \mathtt{x}(\mathtt{t}), \mathtt{y}(\mathtt{t}) >$ und das linke, sowie das rechte Ende eines reellen Intervalls $[a, b]$.
Alle Prozeduren haben auf meinem (relativ Leistungsschwachen) Surface[4] nicht mehr als eine Minute zum Rechnen benötigt, wobei ich aber auch Funktionen und Kurven ausprobiert habe, die nach langem warten[5] gar nichts ausgegeben haben.

0.1 Notation

In diesem Abschnitt werden die in den *Maple*-Skripten verwendeten Notationen und die Farbwahl in den Bildern erläutert. Es ist sinnvoll sich dies zu merken, da wir aus ästhetische Gründen auf eine Legende in den Plots und Animationen verzichtet haben.

Mathematische Objekt	Zeichen	Farbe
Kurve als Vektor	`c(t)`	Grün
Partialfunktionen der Kurve	`x(t), y(t), c(t)[1], c(t)[2]`	-
Intervallränder	`a, b`	-
Laufparameter	`t`	-
Erste Ableitung	`abl, dx, dy`	-
Zweite Ableitung	`abl2, ddx, ddy`	-
Kurvenlänge	`L`	-
Tangente	`v, Tangente`	Cyan
Normale	`n, Normale`	Blau
Krümmung	`kappa, Krümmung, f`	Gelb
Evolute	`Evolute`	Rot
Evolente	`Evolente`	Magenta

Teile des Codes sind in diesem Bericht in `Schreibmaschinenschrift` geschrieben. Mathematische Sachverhalt wie gewohnt *kursiv*.
Wichtige mathematische Dinge, wie Definitionen und Sätze haben ihren eigenen Kasten bekommen. *Maple*-Prozeduren sind durch Striche links und rechts gekennzeichnet, wobei als Schriftart `Schreibmaschinenschrift` verwendet wird.

0.2 Maple

Was uns bei diesem Projekt an dauernd begleitete, sind die Eigenarten von *Maple*. Es macht manchmal einfach, was es will und nicht das, was es tun soll.
Man testet die Berechnung einer Sache in einzelnen Zeilen und ist dann glücklich, wenn es funktioniert. Also kopiert man diese Zeilen in eine Prozedur, kümmert sich um Beginn, Ende und die lokalen Variablen. Anschließend probiert man seine Prozedur aus und *Maple* macht *nichts*.

Scheinbar verhält sich *Maple* inner- und außerhalb von Prozeduren grundlegend anders. In den meisten Fällen haben wir dann einfach so lange rumprobiert und Sachen geändert, bis es irgendwann funktioniert hat[6].
Etwas was ich bis heute nicht verstanden habe, ist wann man *Maple* eine Funktion `c(t)` := ... übergibt und wann nur einen symbolischen Ausdruck in `t` ohne ihn explizit als Funktion zu definieren. Dies ist somit leider auch zwischen den Prozeduren nicht gleich. Mal wurden Funktionen und mal symbolische Ausdrücke verwendet, die dann aber später wieder zu Funktionen umgebaut wurden[7].
Etwas was man auch erstmal wissen muss, um es nicht falsch zu machen, ist, dass man in *Maple* manche griechischen Buchstaben nicht als Variablennamen verwenden darf. Wir versuchten ϑ also `vartheta` als Variable zu verwendet. Außerhalb einer Prozedur war alles in Ordnung, aber in der Prozedur hat er einfach nichts gemacht und leider auch keinen Fehler ausgegeben, was das finden des Problems unheimlich schwierig machte. An anderer Stelle hat das κ beziehungsweise `kappa` für die Krümmung einer Kurve problemlos funktioniert.
Ein weiteres Problem in *Maple* war, dass man Leerzeichen einfach nicht erkennen konnte. Dies lies sich doch relativ leicht beheben, indem man die Schriftart ändert. Alle *Maple*-Skripte sind zur einfacheren Lesbarkeit in der Monospace-Schriftart „Courier New“ geschrieben.

0.3 Quellen

Wir hatten beide noch keine Differentialgeometrie gehört[8]. Das einzige, was uns bekannt war, ist die Formel zur Berechnung der Länge einer Kurve[9].
Die meisten Formeln, die wir verwendeten stammen aus dem Internet. Sehr weiter geholfen hat natürlich *Wikipedia*. Einige Formeln wurden aber auch aus Skripten beziehungsweise der Aufgabenstellung übernommen.
Eine weitere unglaublich wichtige Hilfe war *Google*. Da wir uns beide bisher noch nicht mit *Maple* beschäftigt hatten und der Syntax im Vergleich zu *R* und *Python* ein ganz anderer ist, waren wir gerade am Anfang darauf angewiesen erstmal zu googlen, wie man eine Funktion in *Maple* ableitet oder wie man eine mehrdimensionale Abbildung definiert, plottet und animiert.

1 Kleine Einführung

Wie jeder gute mathematische Text beginnen wir mit folgender

Definition: Eine Kurve ist eine Abbildung mit

$$c : \mathbb{R} \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} c_1(t) \\ c_2(t) \end{pmatrix} \ .$$

Bemerkung:
Eine Kurve ist also eine natürliche Verallgemeinerung eines Graphen. Für diesen gilt

$$f : \mathbb{R} \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} t \\ f(t) \end{pmatrix} \ .$$

Die Ableitungen einer Kurve werden komponentenweise berechnet. Es gilt für die n-te Ableitung mit $n \in \mathbb{N}$

$$c^{(n)} : \mathbb{R} \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} x^{(n)}(t) \\ y^{(n)}(t) \end{pmatrix} \ .$$

Voraussetzung ist, dass die Partialfunktionen $x(t)$ und $y(t)$ n-mal stetig differenzierbar sind.

2 Einfachste Prozeduren

2.1 Plot einer Kurve

Eine Frage, die man sich beim Anblick der Funktiosgleichung einer Kurve natürlich sofort stellt ist: „Wie sieht diese Kurve denn überhaupt aus?“.
Diese Frage beantwortet das erste Skript (*1_Kurve_Plot.mw*). Die Prozedur lautet

```
Kurve_Plot := proc(c, a, b)
plot([c(t)[1], c(t)[2], t = a .. b], color = green);
end proc:
```

Diese Prozedur ist nur zum warm werden. Man übergibt die Kurve `c` und den Intervallanfang `a`, sowie das Intervallende `b`. Die Prozedur plottet dann

den Verlauf der Kurve in einem Ausschnitt der euklidische Ebene.

Eines der wichtigste Beispiele für Kurven ist natürlich der Einheitskreis. Dieser ist gegeben durch

$$c : [0, 2\pi] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} .$$

Ändert man den Definitionsbereich von $[0, 2\pi]$ auf $[0, 2n\pi]$, so wird der Einheitskreis n-mal durchlaufen. Aufgrund der Periodizität von Sinus und Kosinus hat das keinen Einfluss auf den entstehenden Plot. Im folgenden sieht man den Plot des einmal durchlaufenen Einheitskreises.

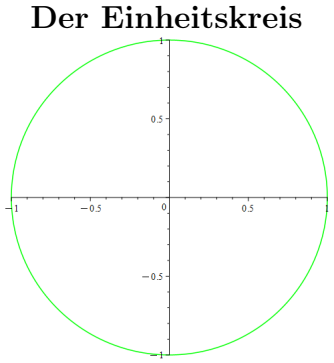

Abbildung 1: Plot von $c(t) = (\cos(t), \sin(t))^\top$ für $t \in [0, 2\pi]$

Das einfügen einer zwei im Sinus führt zu folgender Acht.

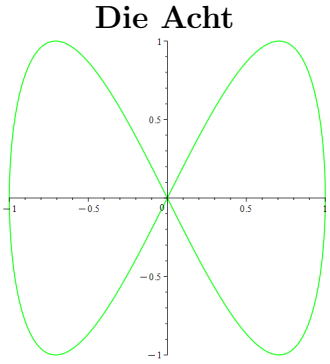

Abbildung 2: Plot von $c(t) = (\cos(t), \sin(2t))^\top$ für $t \in [0, 2\pi]$

Weitere Beispiele und vor allem eigene Experimente lassen sich in *1_Kurve_Plot.mw* machen. Sie haben ja bereits gesehen, was für einen großen Unterschied das Hinzufügen einer einfachen Zwei macht.

2.2 Animation einer Kurve

Jetzt möchte man natürlich nicht nur vor vollendete Tatsachen gestellt werden und ein langweiliges Bild anstarren[10]. Deshalb haben wir eine Animation gebastelt.

```
Kurve_Animation := proc(c, a, b)
local Kurve;
Kurve := proc(t) plot([[c(x)[1], c(x)[2], x = a .. t]], color = green); end proc;
animate([Kurve], [t], t = a .. b, frames = 50);
end proc:
```

Es sei erwähnt, dass für diese Prozedur das Paket `plots` verwendet werden muss.
Innerhalb dieser Prozedur wurde noch eine Prozedur definiert. Die Prozedur `Kurve` gibt einfach für jedes `t` den Plot von `a` bis `t` aus. Für die Länge der Animation haben für 50 Frames gewählt[11].
Ich würde Ihnen jetzt sehr gerne einige Animationen vorführen, aber das ist in LaTeX leider nicht möglich[12]. Die folgende Sammlung von Bildern zeigt eine Animation von

$$c : [0, 2\pi] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \cos(t^2) \\ t \sin(t) \end{pmatrix}$$

bei Frame 30, 40 und 50.

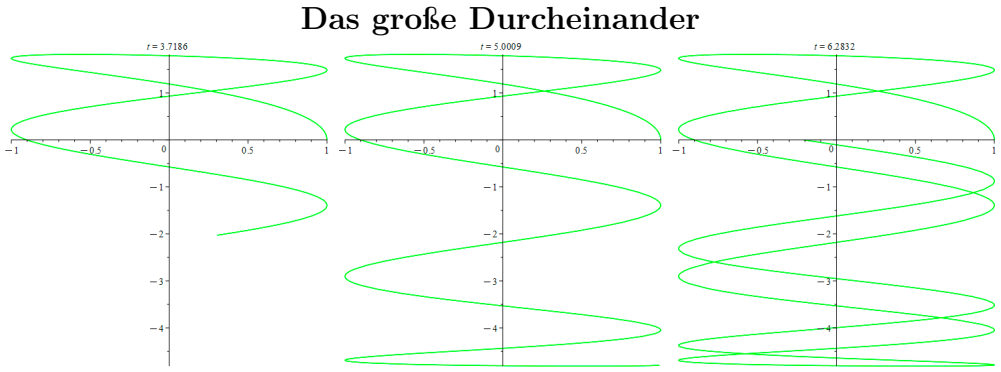

Abbildung 3: „Animation“ von $c(t) = (\cos(t^2), t\sin(t))^\top$ für $t \in [0, 2\pi]$

Einige weitere Animationen[13] lassen sich *2_Kurve_Animation.mw* entnehmen.

2.3 Länge einer Kurve

Nun zu dem Thema, welches mir schon zuvor bekannt war. Die Länge einer Kurve bestimmen. Es gilt die folgende

Definition: Sei $c : [a, b] \to \mathbb{R}^2$ eine stetig differenzierbare Kurve. Die *Länge* der Kurve von a bis b ist definiert als

$$L = \int_a^b \sqrt{x'(t)^2 + y'(t)^2}\,\mathrm{d}\,t\ .$$

Bemerkung:
Diese Formel sieht an sich sehr einfach aus. Doch hier sieht man deutlich das große Problem der Integralrechnung: Nicht jede integrierbare Funktion hat eine geschlossene Formel für die Stammfunktion. Dieses Problem wird uns auch später bei der Rekonstruktion einer Kurve aus ihrer Krümmung begegnen.
Man könnte sich natürlich auch fragen, ob der Integrand $\sqrt{x'(t)^2 + y'(t)^2}$ überhaupt integrierbar ist. Das ist tatsächlich eine gute Frage. Diese ist aber leicht zu beantworten, denn jede stetige Funktion ist auf einem kompakten

Intervall integrierbar.
Nach Voraussetzung sind $x'(t)$ und $y'(t)$ stetig. Die Abbildungen $t \mapsto t^2$ und $t \mapsto \sqrt{t}$ sind ebenfalls stetig. Somit ist auch der Integrand als Linearkombination und Komposition stetiger Abbildungen wieder stetig, also integrierbar.

Nach dieser kurzen Einführung wollen wir mal eine solche Länge von Hand berechnen.
Beispiel:
Wir wollen nun der Einfachheit halber[14] die Länge des Einheitskreises berechnen. Aus der Schule ist bekannt, dass für einen Kreis vom Radius $r > 0$ gelten sollte

$$L = 2\pi r \ .$$

Für den Einheitskreis muss also $L = 2\pi$ sein. Wir wollen das nun nachrechnen. Betrachte dazu

$$\begin{aligned} L &= \int_0^{2\pi} \sqrt{(\cos(t)')^2 + (\sin(t)')^2}\,\mathrm{d}\,t \\ &= \int_0^{2\pi} \sqrt{(-\sin(t))^2 + (\cos(t))^2}\,\mathrm{d}\,t \\ &= \int_0^{2\pi} \sqrt{\sin^2(t) + \cos^2(t)}\,\mathrm{d}\,t \ . \end{aligned}$$

Dank Additionstheorem ist $\sin^2(t) + \cos^2(t) = 1$ für alle $t \in [0, 2\pi]$. Somit gilt

$$\begin{aligned} &= \int_0^{2\pi} \sqrt{1}\,\mathrm{d}\,t \\ &= \int_0^{2\pi} 1\,\mathrm{d}\,t \\ &= [t]_0^{2\pi} = 2\pi - 0 = 2\pi \ .^{15} \end{aligned}$$

Es macht hier einen Unterschied, ob man den Einheitskreis über $[0, 2\pi]$ oder über $[0, 2n\pi]$ definiert. Es kommt entsprechend $L = 2n\pi$ für den n-mal durchlaufenen Einheitskreis raus.
Dies war natürlich ein sehr einfaches Beispiel, aber gleichzeitig auch eines der einzigen, welches sich von Hand mit dem Hauptsatz der Differential- und Integralrechnung und (geschlossenen) Stammfunktionen lösen lässt.

Für genau diese komplizierten Fälle, in denen eine numerische Lösung nötig ist, haben wir die folgende Prozedur geschrieben.

```
Länge := proc(c, a, b)
local t, abl, l, L;
abl := map(diff, c(t), t);
L := evalf(int(norm(abl, 2), t = a .. b));
end proc:
```

Dies ist tatsächlich nichts anderes als das, was wir vorher von Hand berechnet haben.

Für den Einheitskreis spuckt *Maple* auch

$$L = 6.283185308 \approx 2\pi$$

richtig aus.

Wir betrachten nun einen Teil der Neilschen Parabel gegeben durch

$$c : [-2, 2] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} t^2 \\ t^3 \end{pmatrix} \ .$$

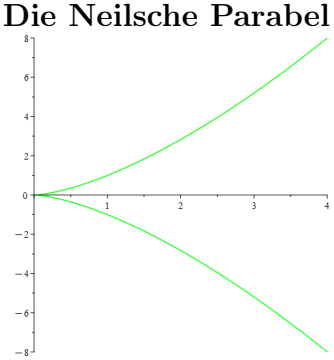

Abbildung 4: Plot von $c(t) = (t^2, t^3)^\top$ für $t \in [-2, 2]$

Lässt man *Maple* mit `Länge(c, -2, 2)` die Länge dieser Kurve berechnen, ergibt das

$$L \approx 18.14683058 \ .$$

Ein Problem tritt bei

$$c : [1, 2] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \tan(t) - 1 \\ t(t^3 - 1) \end{pmatrix}$$

auf. Hier ist *Maple* nicht in der Lage eine Länge zu Berechnen. Nach circa einer Minute an Rechenzeit gibt *Maple* nur

$$L = \int_1^2 \left(\left| 1 + \tan^2(t) \right|^2 + 9 \left| \mathrm{D}(t)(t^3 - 1)^2 \right|^2 |t|^4 \right)^{\frac{1}{4}} \mathrm{d}\, t$$

aus. Trotz dem forcierten `evalf(...)` wird hier kein Wert ausgerechnet[16]. Wir betrachten nun also den Plot von dieser scheinbar so schlimmen Kurve.

Die Kurve ohne Länge

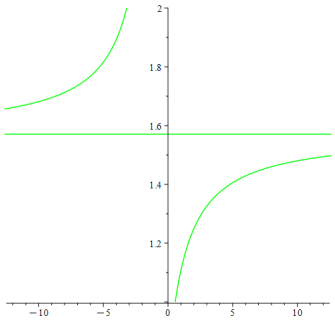

Abbildung 5: Plot von $c(t) = (\tan(t) - 1, t(t^3 - 1))^\top$ für $t \in [1, 2]$

Mit bloßem Auge betrachtet sieht man, dass für die Länge L gelten sollte[17]

$$45 \leq L \leq 50 \ .$$

Wenn man sich die Animation anschaut, sieht man, dass die x-Achsen parallele $y \approx 1.575$ extrem schnell durchlaufen wird. Ich vermute, dass sich die Berechnung des Integrals dort irgendwo aufhängt.

Bei der Berechnung weiterer Beispiele im *Maple*-Skript fällt auf, dass hier teilweise etwas länger für die Werte der Integrale gerechnet wird. Es wird aber (fast) immer ein Ergebnis ausgespuckt.

3 Kompliziertere Prozeduren

Alles bisher präsentierte waren sehr einfache Prozeduren. Nun kommen wir zu etwas komplizierteren, die auch mal etwas an Überlegung benötigt haben.

3.1 Tangente

Die Tangente ist ein normierter Vektor, der die Richtung der Geschwindigkeit der Kurve darstellt.

Definition: Sei $c : [a, b] \to \mathbb{R}^2$ eine stetig differenzierbare Kurve. Die *Tangente* in einem regulären Punkt $t \in [a, b]$ mit $c'(t) \neq 0$ ist definiert als

$$v(t) := \frac{c'(t)}{\|c'(t)\|} \ .$$

Wie immer betrachten wir den Einheitskreis.

Die Tangente an den Einheitskreis

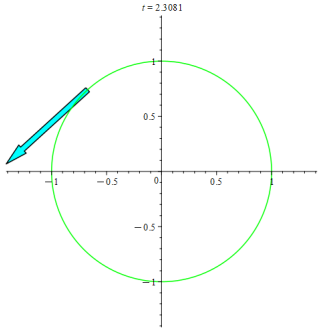

Abbildung 6: Plot der Tangente bei $t \approx 2.31$ an $c(t) = (\cos(t), \sin(t))^\top$

3.2 Normale

Die Normale ist ein normierter Vektor, der senkrecht zur Tangente steht.

Definition: Sei $c : [a, b] \to \mathbb{R}^2$ eine stetig differenzierbare Kurve. Die *Normale* in einem regulären Punkt $t \in [a, b]$ mit $c'(t) \neq 0$ ist definiert

als

$$v(t) := \frac{Jc'(t)}{\|c'(t)\|} \qquad J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} .$$

Wie immer betrachten wir den Einheitskreis.

Die Normale an den Einheitskreis

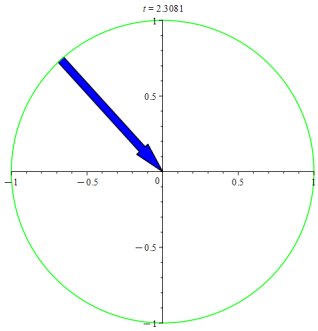

Abbildung 7: Plot der Normale bei $t \approx 2.31$ an $c(t) = (\cos(t), \sin(t))^\top$

3.3 Animation zu Tangente und Normale

Mit den Begriffen und Formeln zu Tangenten und Normalen bewaffnet, können wir uns nun einem der wichtigsten Teile dieses Berichtes widmen.
Wir haben eine Animation gemacht, wie sich die Tangente und Normale in Abhängigkeit von $t \in [a, b]$ auf der Kurve entlang bewegen. Die Prozedur ist gegeben durch[18].

```
Tangente_Normale_Animation := proc(c, a, b)
local abl, J, n, n_1, n_2, v, v_1, v_2, Normale, Tangente, B;
abl := map(diff, c(t), t); J := «0, 1> | <-1, 0»;
n := MatrixVectorMultiply(J, abl)/norm(abl, 2);
n_1 := unapply(n[1], t);
n_2 := unapply(n[2], t);
v := abl/norm(abl, 2);
v_1 := unapply(v[1], t);
v_2 := unapply(v[2], t);
Normale := proc(t) plots[arrow]([c(t)], [n_1(t), n_2(t)], color = blue, symbolsize =
1, color = blue); end proc;
```

```
Tangente := proc(t) plots[arrow]([c(t)], [v_1(t), v_2(t)], color = green, symbolsize =
1, color = cyan); end proc;
B := plot([c(t)[1], c(t)[2], t = a .. b], color = green);
animate([Normale, Tangente], [t, t], t = a .. b, frames = 50, background = B, scaling
= constrained);
end proc:
```

Zuerst einmal ist es hier nötig neben `plots` auch `LinearAlgebra` zu verwenden, um die Matrix-Vektor-Multiplikation und die Normen verwenden zu können.
Auch in dieser Prozedur wurde nichts neues erfunden, sondern einfach die bisher definierten und gefundenen Dinge zusammengebaut.
Der Normalenvektor `n` wird wie in obiger Definition berechnet. Die Schwierigkeit bestand darin, dass *Maple* zum einen die Matrix-Vektor-Multiplikation $\mathtt{J} \cdot \mathtt{n}(\mathtt{t})$ nicht machen wollte und zum anderen auch nicht richtig normierte. Der so gewonnen Vektor wird anschließend in Partialfunktionen zerlegt, damit diese später animiert werden können.
Der Tangentenvektor `v` wird ähnlich berechnet und auch aufgeteilt.
Mit diesen Funktionen werden nun interne Prozeduren `Normale` und `Tangente` definiert, die zu jedem $\mathtt{t} \in [\mathtt{a}, \mathtt{b}]$ den Vektor berechnen, welcher von $\mathtt{c}(\mathtt{t})$ nach $\mathtt{v}(\mathtt{t})$ beziehungsweise $\mathtt{n}(\mathtt{t})$ zeigt.
`B` ist genau der Plot der Kurve, den wir aus dem ersten Teil kennen.
Zum Schluss werden diese Einzelteile durch `animate` in eine schöne Animation überführt.

Auch hier ist es wieder sehr Schade, dass ich Ihnen hier leider keine Animationen zeigen kann. Es sei auf *4_ Tangente_ Normale_ Animation.mw* verwiesen.

Wir betrachten nun ein Kardioid[19] gegeben durch

$$c : [0, 2\pi] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \cos^2(t) + \cos(t) \\ \sin(t)\cos(t) + \sin(t) \end{pmatrix} .$$

In Frame 25, 26 und 40

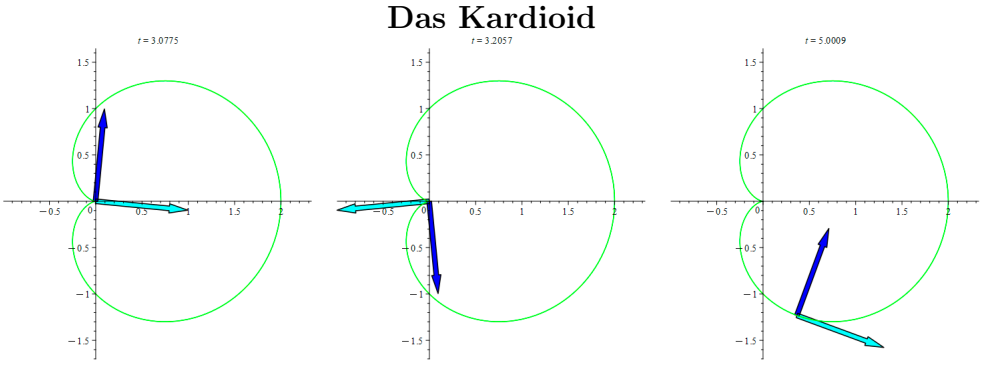

Abbildung 8: „Animation“ der Tangente und Normale von
$c(t) = (\cos^2(t) + \cos(t), \sin(t)\cos(t) + \sin(t))^\top$ für $t \in [0, 2\pi]$

Hier fällt auf, dass *Maple* die Vektoren an der bei π überspringt. In diesem Punkt ist die Ableitung nicht regulär. Deshalb „springen“ die Tangente und die Normale dort.

4 Schwierige Prozeduren

In diesem Teil werde ich einige Prozeduren vorstellen, welche von den Formeln aber vor allem auch durch ihre Implementierung sehr kompliziert waren.

4.1 Krümmung einer Kurve

Definition: Sei $c : [a, b] \to \mathbb{R}^2$ eine zweimal stetig differenzierbare Kurve. Die *Krümmung* der Kurve in einem regulären Punkt ist definiert als

$$\kappa(t) := \frac{\det(c'(t), c''(t))}{\|c'(t)\|^3} \ .$$

Bemerkung:
Für eine nach Bogenlänge parametrisierte Kurve gilt

$$\kappa(t) = \langle c''(t), n(t) \rangle \ .$$

Da die obige Formel allgemeiner ist, haben wir nur diese implementiert.
Weiter ist eine Kurve *rechtsgekrümmt*, falls $\kappa(t) < 0$ ist und *linksgekrümmt*,

falls $\kappa(t) > 0$. Die Art der Krümmung lässt sich den Animationen entnehmen. Positive Krümmungen liegen oberhalb der x-Achse und negative darunter.

Wie vorher wollen wir ein einfaches Beispiel von Hand berechnen.
Beispiel:
Es ist

$$c : [0, 2\pi] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} .$$

Damit ist

$$c'(t) = \begin{pmatrix} -\sin(t) \\ \cos(t) \end{pmatrix} \qquad c''(t) = \begin{pmatrix} -\cos(t) \\ -\sin(t) \end{pmatrix} .$$

Es gilt

$$\det \begin{pmatrix} -\sin(t) & -\cos(t) \\ \cos(t) & -\sin(t) \end{pmatrix} = \sin^2(t) + \cos^2(t) = 1 .$$

Weiter ist

$$\|c'(t)\|^3 = \left\| \begin{array}{c} -\sin(t) \\ \cos(t) \end{array} \right\|^3 = \sqrt{\sin^2(t) + \cos^2(t)}^3 = 1^3 = 1 .$$

Damit ist die Krümmung des Einheitskreises gegeben durch

$$\kappa(t) = 1 .$$

Der Einheitskreis ist also immer gleichstark linksgekrümmt.

In *Maple* haben wir die folgende Prozedur programmiert

```
Krümmung_einer_Kurve := proc(c, a, b)
local abl, abl2, len, kappa, Kurve, Krümmung;
abl := map(diff, c(t), t);
abl2 := map(diff, abl, t);
len := sqrt(norm(abl, 2));
kappa := unapply(Determinant(<abl | abl2>)/len^3, t);
Kurve := proc(t) plot([[c(x)[1], c(x)[2], x = a .. t]], color = green, scaling =
constrained); end proc;
Krümmung := proc(t) plot([[x, kappa(x), x = a .. t]], color = yellow); end proc;
A := Array(1 .. 2);
A[1] := animate([Kurve], [t], t = a .. b, frames = 50);
A[2] := animate([‘Krümmung‘], [t], t = a .. b, frames = 50);
```

```
display(A);
end proc:
```

Die Bestimmung von `kappa` läuft wie in obigem Beispiel. Wir haben uns dann dafür entschieden die Krümmungsfunktion und die Kurve in Abhängigkeit von `t` zu animieren. Deshalb wurden die zwei inneren Prozeduren `Kurve` und `Krümmung` verwendet, was je der Plot vom Zeitpunkt `a` bis zum Zeitpunkt `t` ist. Diese werden zum Schluss nebeneinander ausgegeben.

Die Neilsche Parabel liefert folgenden Plot.

Die geschweifte Klammer

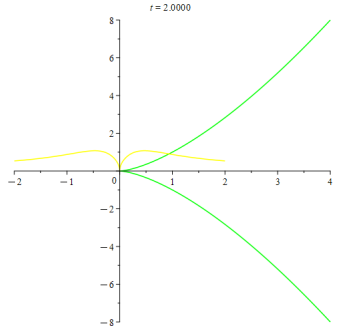

Abbildung 9: Plot der Neilschen Parabel und ihrer Krümmung zu $t \in [-2, 2]$.

Ein weiterer interessanter Plot ist.

Eine interessante Krümmung

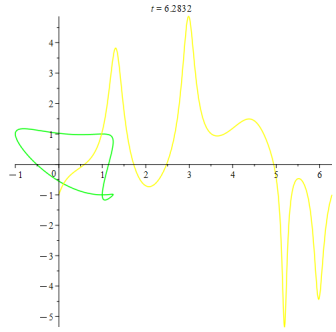

Abbildung 10: Plot der Kurve und Krümmung zu
$c(t) = (\cos(t) + \sin(t)^2, \sin(t) - \cos(t)^3)^\top$ für $t \in [0, 2\pi]$.

Weitere Animationen befinden sich in *5_Krümmung_einer_Kurve.mw*

4.2 Kurve aus Krümmung

Nun stellt sich die Frage der Umkehrbarkeit dieser Operation. Die Antwort liefert der folgende

Satz: Sei $f : [a, b] \to \mathbb{R}$ eine beliebige glatte Funktion, dann existiert eine nach Bogenlänge parametrisierte Kurve $c : [a, b] \to \mathbb{R}^2$, die f als Krümmung hat. Für die Partialfunktionen der Kurve gilt

$$\varphi(t) = \int_v^t f(u)\,\mathrm{d}\,u$$
$$x(t) = \int_v^t \cos(\varphi(u))\,\mathrm{d}\,u$$
$$y(t) = \int_v^t \sin(\varphi(u))\,\mathrm{d}\,u\ .$$

Bemerkung:
Die untere Integrationsgrenze v lässt sich beliebig verändern. Dies führt zu einer Verschiebung der Kurve.

Wie sonst auch wollen wir ein kleines Beispiel von Hand berechnen.
Beispiel:

Sei

$$f : [0, 2\pi] \to \mathbb{R} : t \mapsto 1$$

eine Krümmungsfunktion. Es gilt

$$\begin{aligned}\varphi(t) &= \int_0^t f(u)\,\mathrm{d}\,u \\ &= \int_0^t 1\,\mathrm{d}\,u \\ &= [u]_0^t \\ &= t\ .\end{aligned}$$

Damit ergibt sich

$$\begin{aligned}x(t) &= \int_0^t \cos(\varphi(u))\,\mathrm{d}\,u \\ &= \int_0^t \cos(u)\,\mathrm{d}\,u \\ &= [\sin(u)]_0^t \\ &= \sin(t) \\ y(t) &= \int_0^t \sin(\varphi(u))\,\mathrm{d}\,u \\ &= \int_0^t \sin(u)\,\mathrm{d}\,u \\ &= [-\cos(u)]_0^t \\ &= -\cos(t) + 1\ .\end{aligned}$$

Somit ist die Kurve gegeben durch

$$c : [0, 2\pi] \to \mathbb{R}^2 : t \mapsto \begin{pmatrix} \sin(t) \\ -\cos(t) - 1 \end{pmatrix}\ .$$

Das geübte Auge erkennt hierin den verschobenen Einheitskreis. Dies stimmt also mit dem Ergebnis aus dem vorherigen Kapitel überein.

Die *Maple*-Prozedur lautet.

```
Kurve_aus_Krümmung := proc(f, a, b)
local t, u, phi, x, y, c, Kurve, Krümmung;
phi := int(f(u), u = 0 .. t);
x := int(cos(phi), t = 0 .. s);
y := int(sin(phi), t = 0 .. s);
Kurve := proc(t) plot([[x, y, s = a .. t]], color = green); end proc;
animate([Kurve], [t], t = a .. b, frames = 50);
end proc:
```

Wie immer ist dies einfach die Ausführung der Rechnung aus obigem Beispiel.
Die riesige Schwierigkeit war es hinzubekommen, dass *Maple* hier richtig rechnet. Aus diesem Grund wechseln sich in der Prozedur auch die Integrationsvariablen. Außerdem war es hier wichtig nicht die Funktion `phi(t)` zu integrieren, sondern den Ausdruck `phi`.
Ansonsten haben wir uns hier auch wieder für die bereits bekannte Animation entschieden, denn gerade das folgende Beispiel sieht damit viel besser aus.
Betrachte die Kurve

$$c : [2 - \pi, 2\pi] \to \mathbb{R} : t \mapsto t \ .$$

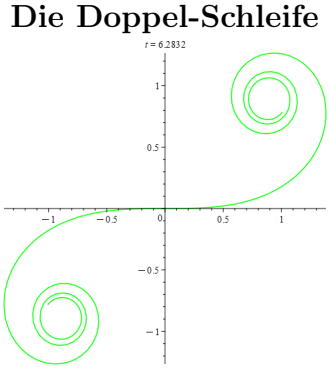

Abbildung 11: Plot zur Kurve aus der Krümmung $f(t) = t$ für $t \in [-2\pi, 2\pi]$.

Es fällt auf, dass die meisten Kurven aus einer Krümmung irgendwelche Schleifen, machen, welche scheinbar auf einen Punkt konvergieren.

Weitere Beispiele dazu befinden sich in *6_Kurve_aus_Krümmung.mw*. Als letztes Beispiel sei

$$f : [-4\pi, 4\pi] \to \mathbb{R} : t \mapsto \sin(t) \ .$$

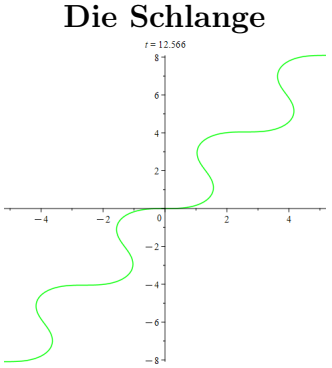

Abbildung 12: Plot der Kurve aus der Krümmung $f(t) = \sin(t)$ für $t \in [-4\pi, 4\pi]$.

Dies ist zwar kein Strudel, aber trotzdem ein interessanter Plot.

Mit diesem Plot möchte ich meinen Bericht beenden. Wir haben noch Programme geschrieben, welche sich mit Evolute und Evolvente beschäftigen. Diese liegen als siebtes und achtes Skript bei[20]. Hier lassen sich auch einige schöne Plots betrachten, aber diese würden diesen nun schon 24 Seiten langen Bericht in meinen Augen sprengen. Es lohnt sich generell einen Blick in die Skripte zu werfen, da sich dort eine sehr große Anzahl an Beispielen befindet, welche diesen Bericht durch die Vielzahl an Bildern nur in die Länge gezogen hätten.

Das arbeiten mit *Maple* und LaTeX hat sehr viel Spaß gemacht[21].

5 Abbildungen

In diesem Teil sind die Kurven und Funktionen zu den jeweils verwendeten Abbildungen. Es ist absolut unnötig diesen Teil zu „lesen“ er ist nur zur Vollständigkeit da, um nachvollziehen zu können, woher die Bilder kommen.

Abbildung 1:
```
c_1 := t -> <cos(t), sin(t)>
Kurve_Plot(c_1, 0, 2*Pi)
```

Abbildung 2:
```
c_2 := t -> <cos(t), sin(2*t)>
Kurve_Plot(c_2, 0, 2*Pi)
```

Abbildung 3:
```
c_5 := t -> <cos(t^2), t*sin(t)>
Kurve_Animation(c_5, 0, 2*Pi)
```

Abbildung 4:
```
c_6 := t -> <t^2, t^3>
Kurve_Plot(c_6, -2, 2)
```

Abbildung 5:
```
c_7 := t -> <tan(t) - 1, t(t^3 - 1)>
Kurve_Plot(c_7, 1, 2)
```

Abbildung 6:
```
c_1 := t -> <cos(t), sin(t)>
Tangente_Normale_Animation(c_1, 0, 2*Pi)
```
Modifiziert durch entfernen von allem, was mit `Normale` zusammenhängt.

Abbildung 7:
```
c_1 := t -> <cos(t), sin(t)>
Tangente_Normale_Animation(c_1, 0, 2*Pi)
```
Modifiziert durch entfernen von allem, was mit `Tangente` zusammenhängt.

Abbildung 8:
```
c_8 := t -> <cos(t)^2 + cos(t), sin(t)*cos(t) + sin(t)>
Tangente_Normale_Animation(c_8, 0, 2*Pi)
```

Abbildung 9:
```
c_6 := t -> <t^2, t^3>
Krümmung_einer_Kurve(c_6, -2, 2)
```

Abbildung 10:
```
c_9 := t -> <cos(t) + sin(t)^2, sin(t) - cos(t)^3>
Krümmung_einer_Kurve(c_9, 0, 2*Pi)
```

Abbildung 11:
```
f_2 := x -> x
Kurve_aus_Krümmung(f_2, -2*Pi, 2*Pi)
```

Abbildung 12:
```
f_7 := x -> sin(x)
```
```
Kurve_aus_Krümmung(f_7, -4*Pi, 4*Pi)
```

6 Fußnoten

[1]Jetzt im Nachhinein fällt mir auf, dass wir das Thema Kurvenscharen scheinbar etwas zu kurz behandelt haben.

[2]Ich fand es zwecks Übersichtlichkeit sinnvoll diese in einzelnen Dateien abzugeben.

[3]Ich bin eine Fußnote.

[4]Surface 7 Pro mit i7 und 8 GB Arbeitsspeicher.

[5]War vermutlich einfach zu kurz.

[6]Somit kann ich Ihnen zu manchen Prozeduren nicht sagen, warum wir das so gemacht haben, wie wir es gemacht haben. Es hat irgendwann einfach funktioniert...

[7]Einer meiner besten Freunde in *Maple* ist die `unapply` Funktion.

[8]Dies wäre im Nachhinein aber sehr sinnvoll gewesen.

[9]Blöderweise war dies nicht einmal Teil der Aufgabenstellung.

[10]Tatsächlich hat es gerade beim Ausprobieren sehr viel Spaß gemacht auf eben diese Bilder zu starren.

[11]Alles andere hat sich zu schnell oder zu langsam angefühlt.

[12]Ich habe tatsächlich eine Anleitung gefunden, aber diese ist viel zu kompliziert.

[13]Und diese sind tatächlich auch animiert.

[14]Alles kompliziertere überlassen wir *Maple*.

[15]Hurra, wir haben richtig gerechnet. Interessant ist, dass LaTeX diesen Teil zweimal als Fußnote ausgibt, obwohl ich ihn nur einmal eingetippt habe.

[15]Hurra, wir haben richtig gerechnet. Interessant ist, dass LaTeX diesen Teil zweimal als Fußnote ausgibt, obwohl ich ihn nur einmal eingetippt habe.

[16]An dieser Stelle war ich etwas enttäuscht, weil bei all meinen Spielereien von Ableitungen über Integrale zu Differentialgleichungen, hat sich *Maple* noch nie geweigert mir ein Ergebnis zu präsentieren.

[17]Gemessen in der Augenmetrik.

[18]Lassen Sie die folgende Prozedur in ihrer vollen Schönheit auf sich wirken.

[19]Oder auf deutsch „Herzkurve“, was ein sehr treffender Name ist, man rotiere einfach die folgende Abbildung um 90 Grad nach rechts.

[20]Sie heißen *7_Evolute.mw* und *8_Evolvente.mw*

[21]Das Wort *Maple* kommt nun insgesamt 33 mal in diesem Text vor.