Objekte und Operationen der mathematischen Mikrowelt „Formel 1“ (Version 2, 03.02.2003)

0 Vorbemerkung

Die neue Version 2 der Mikrowelt erlaubt es, die Nutzeraktivitäten zu speichern, um das Vorgehen später näher zu untersuchen. Ist dies erwünscht, so muss man die Variable „LOGFILE“ zu Beginn der Maple-Sitzung mit einem Dateinamen versehen, etwa:

```
> LOGFILE:="C:\\temp\\log040203";
```

für die Logging-Datei am 4.2.2003. Die Prozeduraufrufe des Benutzers werden an die Logging-Datei „herangehängt“. Später kann man einfach die Logging-Datei laden oder ins Worksheet kopieren und ablaufen lassen.

1 Kurserstellung

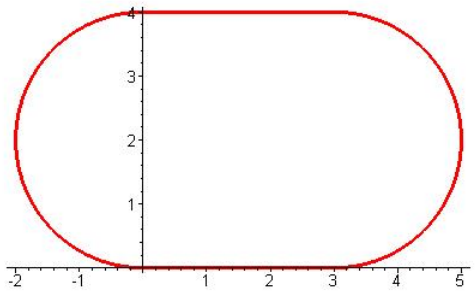
Ein Rennkurs besteht aus Geradenstücken und Kreisstücken. Diese können tangentialstetig aneinandergesetzt werden, d.h. am Zusammensetzpunkt darf kein Knick entstehen (linksseitige und rechtsseitige Tangente müssen übereinstimmen).
Zur Angabe eines Geradenstücks braucht nur dessen Länge angegeben werden. Zur Angabe eines Kreisstücks müssen Länge, Radius und Orientierung (im oder gegen den Uhrzeigersinn) angegeben werden. Um eine gemeinsame Datenstruktur zu verwenden, benutzt man eine Liste mit drei Einträgen:
[Länge, Radius, Orientierung], wobei
- Länge die Länge des Geraden- oder Kreisstückes ist,
- Radius der Radius des Kreisstückes oder bei Geraden „infinity“ (ohne Anführungsstriche!) ist (betrachte Gerade als Kreis mit unendlichem Radius),
- Orientierung entweder `+` oder `-` ist (mit Hochkomma!), je nachdem ob der Kreis mathematisch positiv (also gegen den Uhrzeigersinn) oder mathematisch negativ durchlaufen wird. Bei Geraden ist die Angabe bedeutungslos (wir verwenden immer `+`).

Beispiel:
[3, infinity, `+`] ist eine Gerade der Länge 3;
[2, 1.5, `-`] ist ein Kreisstück der Länge 2 eines Kreises mit Radius 1.5, der im Uhrzeigersinn, also mathematisch negativ durchlaufen wird.
Über die Lage in einem Koordinatensystem ist damit noch nichts ausgesagt, aber das Festlegen eines solchen Systems ist ja auch willkürlich.

Einen Gesamtkurs legt man als Liste solcher Stücke fest, z.B. legt

```
> kursdaten:=[[3, infinity, `+`],[2*Pi, 2, `+`],
>            [3, infinity, `+`],[2*Pi, 2, `+`]];
```

folgenden Kurs fest:



Die Konstruktion erfolgt immer vom Punkt (0,0) ausgehend, womit sich der Benutzer aber nicht zu beschäftigen braucht.
Die Konstruktion solch eines Kurses erfolgt mit der Operation (=Maple-Prozedur)

```
> kurs:=konstruiere_kurs(kursdaten);
```

Der sich ergebende Kurs ist mathematisch eine Kurve in Parameterdarstellung, die stückweise definiert ist (in Maple mit der piecewise-Funktion). Man kann das Ergebnis auch ohne diesen mathematischen Hintergrund nutzen, indem man den Kurs plottet. Dies geht zwar auch direkt mit dem Plot-Befehl in Maple; um aber ein Logging der Benutzeraktion zu realisieren, wird eine eigene Prozedur angeboten, die dieselben Argumente wie das Plotkommando annimmt:

```
> zeige_kurs(kurs);
```

Obiger Plot stellt gerade das Ergebnis dar.
Man kann sich also das Modellieren eines Kurses vorstellen wie etwa das händische Aneinandersetzen von Geraden- und Kurvenstücken einer Carrera-Bahn. Möchte man die Sichtbarkeit der Kursdaten vor dem Benutzer verbergen (etwa weil das geometrische Verständnis nicht vorhanden ist), so kann man auch einen gewissen Satz an Stücken zur Verfügung stellen, den der Benutzer verwenden kann, etwa:

```
> kurze_Gerade:=[ 2, infinity, `+` ]:
> lange_Gerade:=[ 5, infinity, `+` ]:
> krumme_Linkskurve:= [ Pi/4, 1, `+` ]:
> krumme_Rechtskurve:= [ Pi/4, 1, `-` ]:
> schwache_Linkskurve:= [ Pi/4, 2, `+` ]:
> schwache_Rechtskurve:= [ Pi/4, 2, `-` ]:
```

Gegen Ende der Konstruktion ergibt sich häufig das Problem, den Kurs mit Geraden- und Kreisstücken zu schließen. In der realen Situation kann dies mit den gegebenen Stücken unmöglich sein. Läßt man beliebige Kreis- und Geradenstücke zu, so ist dies immer möglich, allerdings mit geometrischem Aufwand verbunden. Will man diesen Aufwand verbergen, so kann man die Operation schliesse_kurs verwenden:

```
> komplette_kursdaten:=schliesse_kurs(kursdaten);
```

Mit dem Resultat (oben gespeichert in „komplette_kursdaten“) kann man dann wieder die Operation konstruiere_kurs ausführen. Gefällt einem die automatisch vorgenommene Komplettierung nicht, so kann man zunächst ein weiteres Stück ansetzen und erneut die Komplettierung durchführen lassen.
Für die Berechnung der Kurslänge steht schließlich folgende Prozedur zur Verfügung:

```
> berechne_laenge(kursdaten);
```

2 Erstellung, Darstellung und Untersuchung von Bewegungsfunktionen und Restriktionen

Um eine Bewegung auf dem Kurs zu beschreiben, kann der Benutzer auf unterschiedlich Weise vorgehen:

a) Definition von s(t)
Hierbei stellt der Benutzer die Funktion s(t) (zurückgelegter Weg in Abhängigkeit der vergangenen Zeit) dar. Die Funktion lässt sich mit den in Maple vorhanden Möglichkeiten im ganzen (als geschlossener Ausdruck) oder stückweise (mit Hilfe der piecewise-Funktion) vorgeben, z.B.:

```
> s:=t->piecewise(t<=3, t, t<=10, t^2+3);
```

Damit einige weitere Prozeduren einwandfrei funktionieren, muss die Funktion genau in der oben angegebenen Weise aufgestellt sein, d.h. in der Form

```
piecewise(t<=Konstante1, Ausdruck1,..., t<=KonstanteN, AusdruckN);
```

Man beachte, dass die Funktion immer stetig sein muss, da das Auto nicht „springen“ kann.
Um solch eine Funktion zu plotten, sollte man die Prozedur

```
> zeige_funktion(s);
```

nutzen, die auch noch weitere Argumente wie bei der plot-Prozedur annimmt. Man könnte natürlich auch schlicht dieplot-Prozedur von Maple verwenden, es wäre dann aber kein Logging möglich. Auch im folgenden sollte also immer, wenn ein Funktionsgrapg zu plotten ist, die Prozedur „zeige_funktion“ verwendet werden!
Will man solch eine Funktion auf dem Kurs ablaufen lassen, so verwendet man die Prozedur

```
> zeige_bewegung(kurs, funktion, bilderzahl);
```

etwa:

```
> zeige_bewegung(kurs, s, 50);
```

Dies erzeugt eine Animation, die einen kleinen Kreis auf dem Kurs ablaufen lässt, wobei die Bilder (mit der jeweiligen Position) für konstante Zeitintervalle erstellt werden und damit einen korrekten Eindruck vermitteln, wenn auch die Ablaufzeit nicht mit der Funktionszeit übereinstimmt.
Eine sinnvolle Bewegungsfunktion kann nur erstellt werden, wenn einzuhaltende Restriktionen bekannt sind, etwa Maximalgeschwindigkeiten in Kurven, Maximalgeschwindigkeit und -beschleunigung (+ und -) des Autos.
Um kursabhängige Restriktionsbedingungen für Geschwindigkeiten leicht formulieren und für den Vergleich heranziehen zu können, wird die Operation

```
> restriktion:=konstruiere_restriktion (kursdaten, maximalwert_liste);
```

angeboten, wobei „kursdaten“ die in 1 beschriebene Liste ist und die maximalwert_liste eine Liste von Maximalwerten auf den einzelnen Kursstücken. Die Liste muss so viele Einträge haben, wie der Kurs Stücke hat. Soll für ein Stück keine Geschwindigkeitsrestriktion gelten, wird einfach „infinity“ eingetragen. So kann man jeden Listeneintrag eindeutig einem Kursstück zuordnen.
Solch eine Restriktionsliste lässt sich veranschaulichen, indem in einem v(s)-Diagramm die Maximalwerte als horizontale Geradenstücke dargestellt werden. Dies erfolgt mit der Prozedur (aus Loggging-Gründen gibt es wieder eine eigene Prozedur!):

```
> zeige_restriktion(restriktion);
```

Hat man nun eine Bewegungsfunktion s(t) erzeugt, so möchte man sehen, ob die Restriktionen eingehalten sind. Dazu werden mehrere Möglichkeiten angeboten:
- Durch Differentiation in Maple kann man v(t) bestimmen. Durch gleichzeitige Darstellung von s(t) und v(t) in einem Plot kann man die Einhaltung der Restriktion überprüfen. Dies ist natürlich mühsam, kann aber auch Einsichten fördern (wohin muss man schauen, um die Einhaltung zu überprüfen?)
- Für die graphische Anzeige wandelt man zunächst s(t) in v(s) um. Dazu dient die Prozedur

```
> vvons:=svont_nach_vvons(s) ;
```
Plottet man jetzt vvons und restriktion gemeinsam (mit „zeige_funktion“ !), so kann man die Einhaltung sehr leicht prüfen.
- Eine Animation unter Restriktionsbedingungen lässt das Auto (also den Kreis) bei überhöhter Geschwindigkeit aus der Kurve hinausspringen. Dazu dient die Prozedur
```
> zeige_bewegung_mit_restriktion(kurs, funktion, restriktion, bilderzahl);
```
Damit kann man schon erkennen, in welchem Bereich man noch weiterarbeiten muss, um eine gültige Fahrfunktion zu erstellen.
Um Beschleunigungen zu überprüfen, kann man die Maple-Möglichkeiten für die zweite Ableitung nutzen.
Um die Güte einer Bewegungsfunktion zu testen, kann man die Rundenzeit ermitteln. Dazu dient die Prozedur:
```
> rundenzeit(kursdaten, funktion);
```
Falls die Runde mit der gegebenen Funktion nicht beendet wird, ergibt sich „unendlich“.
Manchmal möchte man auch zwei Funktionen in der Animation vergleichen (etwa Vergleich mit einen vorgegebenen „guten“ Funktion). Dazu dient die folgende Prozedur, die auch den Vergleich auf einem gewissen Kursabschnitt von Kursstück n zu Kursstück m ermöglicht:
```
> vergleiche_bewegungen(kurs, funktion1, funktion2, n..m, bilderzahl);
```
Da ein Vergleich in einem späteren Kursstück vielleicht gar nicht mehr möglich ist, da ein Auto (=Kreis) dem anderen schon weit voraus ist, ist dies sinnvoll. Gibt man für n..m nun 1..infinity an, so wird der gesamte Kurs betrachtet.

b) Definition von v(t)
Hierbei stellt der Benutzer die Funktion v(t) (Momentangeschwindigkeit in Abhängigkeit der vergangenen Zeit) dar. Die Funktion lässt sich mit den in Maple vorhanden Möglichkeiten im ganzen (als geschlossener Ausdruck) oder stückweise (mit Hilfe der piecewise-Funktion) vorgeben, z.B.:
```
> v:=t->piecewise(t<=3, t, t<=10, t^2+3);
```
Man beachte, dass die Funktion immer stetig sein muss, da die Geschwindigkeit nicht springen kann.
Die Darstellung hat den Vorteil, dass man die Geschwindigkeitsrestriktionen direkt im Griff hat. Problematisch ist allerdings, dass die Restriktionen weg- und nicht zeitbezogen sind (Maximalgeschwindigkeit an einer Stelle, nicht zu einer Zeit!).
Will man die resultierende Weg-Zeit-Funktion ermitteln, so nutzt man einfach die Integrationsmöglichkeiten von Maple, wobei die Anfangsbedingung s(0)=0 zu beachten ist, z.B.
```
> int(v(t),t);
```
Mit dem so gewonnenen s(t) können dann alle im Abschnitt a) genannten Darstellungs- und Untersuchungsmöglichkeiten genutzt werden. Man beachte aber, dass s(t) bei piecewise-Funktionen in einigen Mikroweltprozeduren genau die oben in a) angegebene Struktur haben muss (gegebenenfalls muss man den letzten Teilausdruck „abschneiden“).

c) Definition von v(s)
Hierbei stellt der Benutzer die Funktion v(s) (Momentangeschwindigkeit in Abhängigkeit der betrachteten Stelle) dar. Die Funktion lässt sich mit den in Maple vorhanden Möglichkeiten im ganzen (als geschlossener Ausdruck) oder stückweise (mit Hilfe der piecewise-Funktion) vorgeben, z.B.:
```
> v:=s->piecewise(t<=4, sqrt(t), t<=10, 2);
```
Man beachte, dass die Funktion immer stetig sein muss, da die Geschwindigkeit nicht springen kann.
Die Darstellung hat den Vorteil, dass man die wegbezogenen Geschwindigkeitsrestriktionen direkt im Griff hat, man kann deren Einhaltung sofort überprüfen, indem man etwa Funktion und Restriktionen (siehe a)) zusammen plottet:

```
> restriktion:=konstruiere_restriktion (kursdaten, maximalwert_liste);
> zeige_funktion({restriktion,v(s)}, s=0..smax);
```
Problematisch ist hierbei, dass man nicht unmittelbar die Weg-Zeit-Funktion aus v(s) ermitteln kann. Eigentlich ist v(s)="Ausdruck in s" wegen $v = \dot{s}$ eine Differentialgleichung für s(t) mit Anfangsbedingung s(0)=0. Um diese mathematische Thematik verbergen zu können, wird eine Umwandlungsfunktion angeboten:
```
> svont:= vvons_nach_ svont(v) ;
```
Mit dem so gewonnenen svont(t) können dann alle im Abschnitt a) genannten Darstellungs- und Untersuchungsmöglichkeiten genutzt werden. Es ist aber zu beachten, dass nicht für alle Funktionen v(s) ein s(t) existiert, das die Anfangsbedingungen s(0)=0 und v(0)=0 erfüllt (Start am Anfang aus ruhendem Zustand) und nicht die Nullfunktion ist. Dies erkennt man, wenn man aus v(s) mit Kettenregel die Beschleunigung a ermittelt:

$$a := \frac{dv}{dt} = \frac{dv}{ds} \cdot \frac{ds}{dt} = \frac{dv}{ds} \cdot v \quad (*)$$

Wenn man nun eine Funktion v(s) konstruiert, für die v(0)=0 und $\frac{dv}{ds}(0) = p \in \mathbb{R}$ mit konstanten p gilt (etwa bei v(s)=s, also v(0)=0, $\frac{dv}{ds} = 1$, also $\frac{dv}{ds}(0) = 1 = p \in \mathbb{R}$), so folgt aus Gleichung (*) für die Beschleunigung a(0)=0*p=0. Wenn am Anfang aber Geschwindigkeit und Beschleunigung gleich 0 sind, so kommt das Auto „nicht vom Fleck". Man sieht hier schon, dass wegen v(0)=0 als notwendige Bedingung für eine positive Beschleunigung an der Stelle 0 zu prüfen ist: $\lim_{s \to 0} \frac{dv}{ds}(s) = \infty$. Dies kann man auch mit Maple machen:
```
> limit(diff(v(s),s),s=0);
```
Genauer kann man auch das in (*) dargestellte Produkt betrachten:
```
> limit(diff(v(s),s)*v(s),s=0);
```
Hier muss sich ein positiver Wert ergeben, damit wirklich losgefahren wird.
Will man diese Grenzwertbetrachtung vermeiden, so kann man die Prozedur
```
> check_vvons(v);
```
verwenden, die eine Überprüfung vornimmt und als Ergebnis "ok" oder "not ok" liefert.

d) Definition von a(s)
Hierbei stellt der Benutzer die Funktion a(s) (Momentanbeschleunigung in Abhängigkeit der betrachteten Stelle) dar. Die Funktion lässt sich mit den in Maple vorhanden Möglichkeiten im ganzen (als geschlossener Ausdruck) oder stückweise (mit Hilfe der piecewise-Funktion) vorgeben, z.B.:
```
> a:=s->piecewise(t<=4, t, t<=10, 2);
```
Was Sprungmöglichkeiten betrifft, kann man hier etwas „großzügiger" sein und diese zulassen. Man modelliert dann ein theoretisch unendlich schnelles Durchtreten des Pedals (oder Vollgas per Kippschalter). Man kann dies aber auch untersagen und nur einen exponentiellen Verlauf (Sättigungsfunktion) zum Maximalwert zulassen.
Die Darstellung hat den Vorteil, dass man Beschleunigungsrestriktionen direkt im Griff hat.
Problematisch ist wiederum, dass man nicht unmittelbar die Weg-Zeit-Funktion aus a(s) ermitteln kann. Eigentlich ist a(s)="Ausdruck in s" wegen $a = \ddot{s}$ eine Differentialgleichung 2. Ordnung für s(t) mit Anfangsbedingungen s(0)=0 und v(0)=0. Um diese mathematische Thematik verbergen zu können, wird eine Umwandlungsfunktion angeboten:
```
> svont:= avons_nach_ svont(v) ;
```
Mit dem so gewonnenen svont(t) können dann alle im Abschnitt a) genannten Darstellungs- und Untersuchungsmöglichkeiten genutzt werden. Es ist aber zu beachten, dass a(0) positiv sein muss, damit sich das Auto bewegt.

3) Tabelle der Objekte und Operationen

Kurs und Kurserstellung	**Kommentar / Beispiel**
kursstueck:=[laenge, radius, orientierung];	s1:=[3,infinity,`+`]; Geradenstück der Länge 3; k1:=[3,5,`-`]; Kreisstück mit Länge 3, Radius 5, im Uhrzeigersinn durchlaufen.
kursdaten:=[stueck1,stueck2,...,stueckn];	kursdaten:= [[3,infinity,`+`],[3,5, `-`]]; oder kursdaten:=[s1,k1];
kurs:=konstruiere_kurs(kursdaten);	
zeige_kurs(kurs);	Plottet den Kurs
komplette_kursdaten:=schliesse_kurs(kursdaten);	
laenge:=berechne_laenge(kursdaten);	Gibt die Kurslänge an.
Restriktionen	
restriktion:= konstruiere_restriktion(kursdaten, maximalwert_liste);	maximalwert_liste:= [10,20,infinity,10,infinity,20];
zeige_restriktion(restriktion);	zeigt eventuell lückenhaftes v(s)-Diagramm für Restriktionen
Fahrfunktionen	
Umwandlungsprozeduren:	
svont_nach_vvons(s) ;	Input und Output sind Funktionen, nicht Ausdrücke!
vvons_nach_svont(v);	dito.
avons_nach_svont(a);	dito.
Funktionsuntersuchung:	
zeige_funktion(s);	Plottet die Funktion.
rundenzeit(kursdaten, funktion);	Berechne Rundenzeit für eine Bewegungsfunktion bei gegebenem Kurs
check_vvons(v);	Prüfe, ob v(s) notwendige Bedingung erfüllt
Animation	
zeige_bewegung(kurs, funktion, bilderzahl)	Zeigt Ablauf eines Kreises auf dem Kurs gemäß der Bewegungsfunktion, z.B. zeige_bewegung(kurs,s,20);
zeige_bewegung_mit_restriktion(kurs, funktion, restriktion, bilderzahl)	Bezieht Restriktion mit ein; bei Nichteinhaltung springt der Kreis aus der Bahn
vergleiche_bewegungen(kurs, funktion1, funktion2, n..m, bilderzahl);	Läßt zwei Kreise gemäß den beiden Funktionen laufen, und zwar auf Streckenabschnitt n bis m (Kursstück m bis Kursstück n); ist n..m=1..infinity, so wird der ganze Kurs durchlaufen.

4 Mathematische und physikalische Kenntnisse für die Arbeit in der Mikrowelt

Die notwendigen Kenntnisse zur sinnvollen Benutzung der Mikrowelt hängen davon ab, welche kompakten Prozeduren zugelassen werden (siehe nächster Abschnitt). Damit kann man das Ausmaß der erforderlichen Kenntnisse steuern. Im Folgenden geben wir alle mathematischen und physikalischen Themen an, die angesprochen werden könnten. Kenntnis bedeutet hier nicht unbedingt tiefgehendes Verständnis. Eine tiefere Durchdringung der Materie soll gerade im Rahmen der Behandlung in der Mikrowelt erfolgen.

4.1 Physikalische Kenntnisse

Da es in der Mikrowelt um mathematische Beschreibungen von Bewegungsvorgängen geht, sollten die wesentlichen Begriffe zur eindimensionalen Bewegungsbeschreibung und deren Zusammenhänge bekannt sein, genauer:

- Beschreibung einer eindimensionalen Bewegung durch eine Weg-Zeitfunktion s(t)
- Geschwindigkeit-Zeit- und Beschleunigung-Zeit-Funktion v(t) bzw. a(t):

$$v(t) = \dot{s}(t) = \frac{ds}{dt}(t), \quad a(t) = \dot{v}(t) = \frac{dv}{dt}(t), \quad a(t) = \ddot{s}(t).$$

- Ein nicht beschleunigter Körper bewegt sich mit konstanter Geschwindigkeit.

Als Vereinfachung wird nur das eindimensionale Problem (also eigentlich nur Tangentialgeschwindigkeit und –beschleunigung) betrachtet, obwohl der Kurs eigentlich zweidimensional ist.

4.2 Mathematische Kenntnisse

Für die Kurserstellung sind folgende Kenntnisse über Kreise erforderlich

- Ist α der Winkel eines Kreissektors in Bogenmaß(!) und r der Radius, so ist die Länge l des Kreisbogens aus $l=\alpha*r$ zu ermitteln.
- Kennt man zwei Punkte P1, P2 und den Radius eines Kreises und weiß man, auf welcher Seite der Geraden P1P2 der Mittelpunkt liegt, so kann man die Länge des Kreisbogens vom einen Punkt zum anderen bestimmen (wenn ein gewisser Durchlaufsinn gegeben ist).

Behandelt man auch das Kursschließungsproblem, so sollten Kenntnisse über geometrische Konstruktionen von Winkelhalbierenden und Senkrechten sowie von tangentialer Lage von Gerade und Kreis vorhanden sein.
Will man tiefer in die mathematische Beschreibung eines Kurses einsteigen, so benötigt man den Begriff der Kurve in Parameterdarstellung und entsprechende Darstellungen für Geraden (normalerweise im Rahmen der Vektorrechnung behandelt) und Kreise.

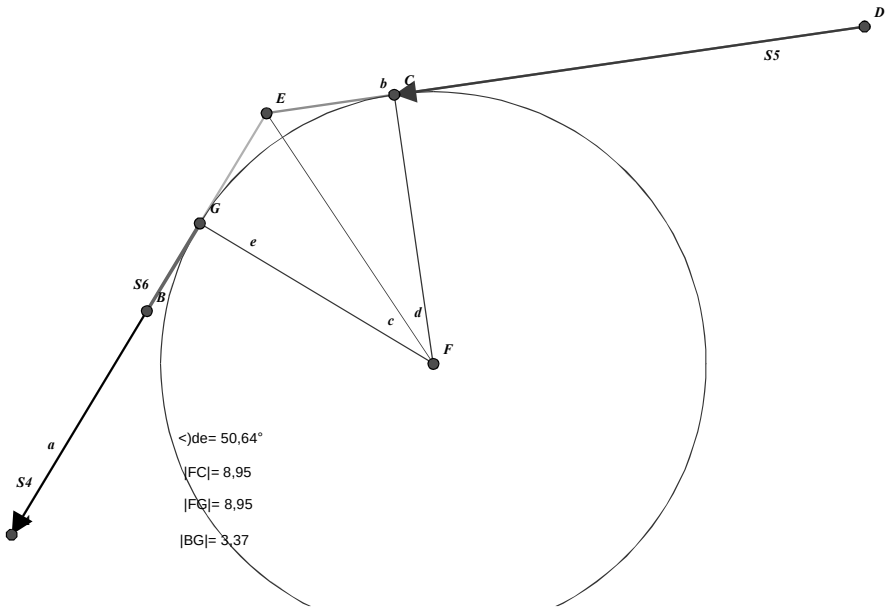
Für die Erstellung und Behandlung der Funktionen sollte zunächst einmal ein gewisser Fundus an Funktionsklassen vorhanden sein (Polynome, Wurzeln, gebrochenrationale, Exponentialfunktionen). Ferner sollten die Begriffe Stetigkeit und Differenzierbarkeit bekannt sein. Die stückweise Definition von Funktionen ist im Prinzip kein neuer Funktionsbegriff und könnte im Rahmen der Mikrowelt eingeführt werden.
Um zwischen s(t), v(t) und a(t) zu wechseln, sind Differentiation und Integration erforderlich. Will man die Berechnung von s(t) aus v(s) oder a(s) genau nachvollziehen, so benötigt man die Integration (oder man geht den allgemeinen Weg über Differentialgleichungen).

5 Aufgaben und Lernwege in der Mikrowelt

Im Folgenden werden einige Möglichkeiten beschrieben, Lernaktivitäten in der Mikrowelt zu initiieren. Mathematische Themen und Vorgehensweisen können sichtbar oder in Prozeduren verborgen sein. Ob man eine Prozedur nutzt, die Mathematik verbirgt, hängt von den jeweiligen Lernvoraussetzungen und Lernzielen ab.

5.1 Kurserstellung

- Will man sich gar nicht mit der „Geometrie“ der Kurserstellung beschäftigen, so kann man einen Kurs einfach vorgeben. Dann finden auf dieser Ebene keine Lernaktivitäten statt, man konzentriert sich gänzlich auf die Funktionen. Dies ist nicht unproblematisch, da zumindest eine Kenntnis der stückweisen Zusammensetzung des Kurses auch als Hintergrund für die stückweise Funktionsbetrachtung und –erstellung wichtig ist.
- Man kann vordefinierte Stücke (Kurzes_Geradenstueck usw.) anbieten und als Aufgabe stellen, damit einen Kurs mit gewissen Eigenschaften zu konstruieren (z.B. mindestens ein Wechsel des Krümmungsvorzeichens, keine Überschneidung, ...). Hier wird dann schon das Problem auftauchen, wie man den Kurs mit den vorhandenen Mitteln schließen kann. Geht dies nicht, so kann man die Schließ-Prozedur nutzen (und eventuell untersuchen, was dort eigentlich gemacht wird).
- Man kann die Definition von Stücken anbieten und wiederum als Aufgabe stellen, damit einen Kurs mit gewissen Eigenschaften zu konstruieren. Hier ist zunächst eine Beschäftigung mit der Frage erforderlich, durch welche „Daten“ denn ein Stück bzw. sein Einbau in den Kurs eindeutig beschrieben wird. Beispielsweise kann ja in einem Carrera-Kurs ein- und dasselbe Kurvenstück als Linkskurve oder als Rechtskurve gesteckt werden, also benötigt man eine Orientierungsinformation. Auch hier taucht wieder das Problem der Kurs-Schließung auf, das behandelt oder verborgen werden kann.
- Eine Behandlung des Kurs-Schließungsproblems führt zu interessanten geometrischen Fragen und Konstruktionen, etwa:
  - Mit wie vielen Stücken kommt man immer aus?
  - Wie kann man in verschiedenen Situationen diese Stücke konstruieren?
  Um zu sehen, dass man nicht mit einer Konstruktion auskommt, kann man auch verschiebbare Konfigurationen in einem Dynamischen Geometrie-System (DGS) betrachten. Die unten abgebildete Konfiguration wurde für diesen Zweck entworfen: Bestimmen der Vektor BA Anfang und Anfangsrichtung und der Vektor DC Ende und Endrichtung, so ergibt die Konstruktion ein Kreisstück (zwischen C und G) und ein Geradenstück (zwischen G und B), die zusammen den Kurs schließen. Bewegt man den Punkt C, so sieht man, dass die Konstruktion nicht immer zum gewünschten Ergebnis führt. Dies kann wesentlich zur Erkenntnis beitragen, dass das Arbeiten mit einer konkreten Konfiguration kein vollständiger Beweis ist, sondern eine Klassifikation möglicher Situationen erforderlich ist.
- Man kann auch die Nachbildung eines realen Rennkurses (z.B. Hockenheimring) als Aufgabe stellen, wobei die Stücke dann wieder frei definiert werden. Angaben über Kreis- und Geradenstücke sind im Internet zu finden (z.B. www.hockenheimring.de). Solch eine Aufgabe hat den Reiz der Realitätsnähe, ist aber auch deutlich aufwändiger. Daher ist es (allerdings abhängig von den Lernvoraussetzungen) wohl in der Regel sinnvoller, mit einem kleinen selbsterstellten Kurs anzufangen. Dies kann auch die Nachbildung einer Carrera-Bahn sein.



5.2 Fahrfunktionen

Bei der Aufstellung von Fahrfunktionen bieten sich verschiedene Zugänge an, die sich in Bezug auf Einfachheit und Nähe zu möglichen Restriktionen unterscheiden. Sinnvolle mögliche Restriktionen sind Geschwindigkeitsbeschränkungen für gewisse Streckenabschnitte und Maximalgeschwindigkeit und –beschleunigung (letztere positiv und negativ). Naheliegend ist es, zur eindeutigen Bewegungsbeschreibung zunächst eine Weg-Zeit-Funktion s(t) aufzustellen, zu untersuchen und entsprechend zu ändern. Da sich die Restriktionen im wesentlichen auf Geschwindigkeiten beziehen, kann man aber auch mit der Aufstellung eine Geschwindigkeit-Zeit-Funktion v(t) oder, da die Restriktionen wegbezogen sind, mit einer Geschwindigkeit-Weg-Funktion (wo wie schnell?) v(s) beginnen. Ein weiterer Einstieg ist über die Beschleunigungsrestriktionen möglich, also über a(t) oder a(s).
Ein - vielleicht erst späteres - Durchspielen solcher Möglichkeiten zeigt, dass Abhängigkeiten unterschiedlich modelliert werden können und dass Variablen wie z.B. s in unterschiedlichen Rollen (mal abhängige, mal unabhängige Variable) auftreten können, je nach dem betrachteten funktionalen Zusammenhang.
Im Folgenden umreißen wir die Zugänge über s(t) und v(s).

a) Definition einer Weg-Zeit-Funktion s(t)

- Ein einfacher Einstieg kann über eine zunächst restriktionsfreie Betrachtung von einfachen s(t)-Funktionen erfolgen, die als geschlossene Ausdrücke aufgeschrieben werden können, etwa s(t)=t^2. Solche Funktionen können in der Animation dargestellt und miteinander verglichen werden und man kann die Rundenzeiten ermitteln. Dies liefert einen einfachen Einstieg. Zu überprüfen wäre, ob die aufgestellte Funktion den Anfangsbedingungen (Start aus der Ruhelage, s(0)=0, v(0)=0) entspricht.
  Selbst in dieser einfachen Situation stellen sich schon interessante Fragen beim „Wettbewerb“ der Funktionen:
  - Welche Funktion hat das beste Wachstum, t^2, t^3, t^4,..., exp(t), a^t,...?
  - Wie kann man das Wachstum steigern (einfach: konstanter Faktor, Exponent erhöhen)?

- Welche Funktion ist auf welchem Strecken- oder Zeitabschnitt besonders gut (Exponentenerhöhung verschlechtert z.B. zwischen 0 und 1, verbessert aber ab 1)? Dies könnte schon zu der Idee führen, abschnittsweise zu arbeiten.

- Die Einführung von Restriktionen kann bei realen Kursen auf realen Daten (etwa aus dem Internet) beruhen, bei frei konstruierten sind sie mehr oder weniger willkürlich (man könnte sich hier vielleicht an Restriktionen für Carrera-Bahnen orientieren). Wir beschränken uns hier auf Geschwindigkeitsrestriktionen (später könnte die Frage auftauchen: „Von 0 auf 100kmh in 0.1s, geht denn das?“, was zu Beschleunigungsrestriktionen führt). Restriktionen können vom Nutzer als Datenstruktur definiert werden oder sie sind vorgegeben und der Benutzer verwendet sie nur zum Check.
  Man kann jetzt zunächst mit einfachen Funktionen experimentieren, indem man die Einhaltung der Restriktionen prüft. Dies geschieht optisch mit Hilfe der Animation oder prozedural durch
  - Bilden von v(t) (diff(s(t),t)), Plotten und optische Prüfung (aber: Wo soll man hinschauen bei v(t) statt v(s)?). Will man den Ort und die Zeit der „Verletzung“ genauer bestimmen, so kann man auch zunächst mit der Animation den Ort ungefähr bestimmen und mit s(t) die Zeit. Dann sieht man mit v(t), welche Restriktionsgeschwindigkeit v_0 überschritten wurde und ermittelt mit $v(t)=v_0$ die zugehörige Zeit.
  - Umwandlung in v(s) und Vergleich mit der Restriktionsfunktion (gemeinsamer Plot); für die Umwandlung kann die Prozedur svont_nach_vvons verwendet werden, wenn die Umrechnung verborgen werden soll oder zu zeitaufwendig ist (etwa bei stückweise definierten Funktionen), oder es ist gerade ein Lernziel, zu erkennen, wie man aus bestehenden Abhängigkeiten andere ermittelt (bilde v(t) und t(s) (umkehrbar?), setze ein).
  Das Anpassen an Restriktionen führt sehr wahrscheinlich zur Frage der stückweisen Definition von Funktionen, da man kaum eine Funktion findet, die durch einen geschlossenen Ausdruck gegeben ist, alle Restriktionen erfüllt und nicht ganz offenbar „schlecht“ ist. Selbst wenn man etwa an eine lineare Funktion s(t)=a*t+b denkt, wobei a kleiner gleich der stärksten Geschwindigkeitsbeschränkung ist, ergibt sich das Problem, dass die Anfangsbedingung v(0)=0 verletzt ist.
  Hat man den Ort und die Zeit der Geschwindigkeitsverletzung gefunden, so kann man von dort an etwa mit einem Geradenstück weitermodellieren. Auf jeden Fall stellt sich die Frage der Anschlussbedingungen zweier Stücke; hier kommen also ganz natürlich Fragen der Stetigkeit und Differenzierbarkeit auf. Hat man einen nicht-stetigen Anschluß, so ergibt sich beispielsweise in der Animation ein Sprung. Hat man einen nicht-differenzierbaren Anschluß, so ergibt sich in der v(t) und in der v(s)-Funktion ein Sprung.
  Hat man ein Stück modelliert, in dem eine gewisse Restriktion eingehalten wird, ergibt sich natürlich die Frage, ab wann man wieder schneller fahren kann. So kann man sich von einem Stück zum nächsten „vorhangeln“.

- Ist eine Funktion gefunden, die die Restriktionen einhält, stellt sich die Frage der Güte und des Vergleichs von Lösungen. Hierzu können mehrere Wege gegangen werden:
  - Pauschal geht es zunächst um Rundenzeiten, die ermittelt und verglichen werden.
  - Ein Vergleich von v(s) mit der Restriktionsfunktion zeigt, in welchen Bereichen noch „Luft“ ist.
  - Wenn man langsamer ist, möchte man wissen, auf welchem Stück man Zeit gegenüber dem „Wettbewerber“ verliert. Hierzu kann man die Bewegungen in einer gemeinsamen Animation zeigen und auch jeweils Durchläufe eines kleineren Abschnitts vergleichen, um genauere Information zu bekommen.

b) Definition einer Geschwindigkeit-Weg-Funktion v(s)
Da sich die Restriktionen auf die maximalen Geschwindigkeiten beziehen, liegt auch die Idee nahe, eine Geschwindigkeit-Weg-Funktion v(s) aufzustellen. Diese kann dann gleich mit den Restriktionen verglichen werden, man kann sogar die Restriktionen direkt bei der Aufstellung der Funktion benutzen. Da die Restriktionen stückweise definiert sind, liegt eine stückweise Aufstellung der v(s)-Funktion nahe. Hierbei tauchen folgende Probleme auf:

- Damit sich schlussendlich eine Bewegung ergibt, die vom Start wegführt, muss die Ableitung nach s in 0 unendlich sein, die Funktion muss also mit vertikaler Tangente hochlaufen (siehe Abschnitt 2c). Wählt man also diesen Zugang, so sollte mit den Benutzern diese Anforderung gemäß den Ausführungen in 2c) geklärt werden. Dieser Punkt führt natürlich auch zur interessanten Frage, welche Funktionen denn die Anforderung erfüllen und wie man solche Funktionen konstruieren kann. Neben n-ten Wurzeln kommen hierbei nach den Ableitungsregeln auch Funktionen wie $\sqrt{x}+x$ oder $\sqrt{x}\cdot\sin(x)$ in Frage. Der zunächst naheliegende Ansatz, die lückenhafte Restriktionsfunktion einfach durch lineare Verbindungsstücke zu vervollständigen, ist am Anfang nicht zulässig, da lineare Funktionen (etwa f(x)=2*x) die Anforderung nicht erfüllen. Man kann sich hier auch die Frage stellen, von welcher Art von Weg-Zeit-Funktion sich denn solche v(s)-Funktionen ergeben (z.B. ergeben quadratische Polynome bei s(t) konstante Beschleunigungen und bei v(s) Wurzelfunktionen).
- Was das Zusammensetzen von einzelnen Funktionsabschnitten betrifft, so treten dieselben Fragen bezüglich Stetigkeit und Differenzierbarkeit auf wie in a). Stetigkeit ist wieder zu fordern, weil die Geschwindigkeit sich nicht sprunghaft erhöhen kann, Differenzierbarkeit braucht man nicht zu fordern, wenn man Beschleunigungssprünge zulässt.

Ist eine v(s)-Funktion konstruiert, so stellt sich die Frage, welche Bewegungsfunktion s(t) daraus resultiert und wie letztendlich die Rundenzeit ist. Nun handelt es sich bei der Funktionsgleichung v(s)="Ausdruck in s" wie in 2c) beschrieben um eine Differentialgleichung, die zur Bestimmung von s(t) zu lösen ist. Man kann dies allerdings auch auf die Berechnung eines Integrals reduzieren. Will man diese Thematik nicht behandeln und insbesondere nicht die aufwändige Bearbeitung stückweiser Funktionen bei solchen Problemen betrachten, so kann man an dieser Stelle die Umwandlungsprozedur vvons_nach_svont verwenden. Dann hat man wieder die in a) beschriebenen Untersuchungs- und Animationsmöglichkeiten.
Betrachtet man zusätzlich positive und negative maximale Beschleunigungen des Autos und lässt man Beschleunigungssprünge zu, so lässt sich sogar eine optimale Funktion konstruieren, indem solange Vollgas gegeben wird, bis man die maximale Geschwindigkeit des Autos erreicht hat oder bis man bremsen muss, um rechtzeitig zur nächsten Restriktion auf die erforderliche Geschwindigkeit abgebremst zu haben.