Mathematik und Wirklichkeit
- zwei Fallbeispiele

A. P. Barth, O. Eglin, J. Ferrari, F. Müller, M. Tschabold

Es scheint, dass es mindestens drei verschiedene Gründe gibt, weshalb man Mathematik treiben kann: Man kann das Ziel verfolgen, Ausschnitte der Welt zu *erklären*, oder das Ziel, Ausschnitte der Welt zu *beschreiben*. Und schliesslich kann man die Mathematik auch um ihrer selbst Willen lieben. Was wir hier vorstellen, ist ein kleiner Beitrag zum zweiten Aspekt. Wir zeigen den Einsatz gymnasialer Mathematik bei der Beschreibung von zwei Weltausschnitten, nämlich den Einsatz von Spline-Funktionen bei der Modellierung einer Auto-Karosserie und den Einsatz von Vektorgeometrie bei der Prognose eines Raketenfluges.

I. Hintergrund

An unserem Gymnasium wird während der letzten beiden Jahre das Schwerpunktfach *Mathematik und Anwendungen der Physik* mit sechs Lektionen pro Woche angeboten, ein akzentuierendes Fach, das aus einer Reihe von Schwerpunktfächern gewählt werden kann. In einem solchen Kurs haben wir uns während eines Semesters unter anderem mit den hier vorgestellten Themen beschäftigt. Wir: Das sind ein Mathematiklehrer, eine Schülerin und drei Schüler. Die von uns erarbeiteten Themen erheben nicht den Anspruch auf Neuheit, doch es scheint uns, sie eignen sich für den Einsatz im Gymnasium aus mehreren Gründen besonders gut: Zum einen sind sie relativ aktuell und anwendungsorientiert, zum anderen bieten sie die Möglichkeit, einen modernen CAS-Rechner und auch eine starke Mathematik-Software (in unserem Fall MATLAB) gewinnbringend einzusetzen. Schliesslich machen sie eindrücklich klar, dass die oft etwas theoretisch anmutenden Stoffe des Grundlagenunterrichtes in Mathematik vielfältigen praktischen Zwecken dienen können.

Im Kapitel II stellen wir kurz Geschichte und Theorie der Spline-Funktionen vor, um dann zu einem typischen Praxiseinsatz solcher Funktionen überzugehen, dem Computer-Design der Karosserie eines Wagens, in unserem Fall eines BMW 745i. In Kapitel III lassen wir eine kleine selbstgebastelte Rakete steigen, erheben Flugdaten messtechnisch und vergleichen diese mit einem auf dem Taschenrechner TI-92 programmierten mathematischen Modell der Flugbahn. Während sich die erste Anwendung schon kurz nach Erarbeitung der Grundlagen der Differentialrechnung untersuchen lässt, sehen wir eine Behandlung der zweiten Anwendung sogar früher, sobald etwas Vektorgeometrie behandelt worden ist; zudem baut diese zweite Anwendung eine willkommene Brücke zur Physik.

II. Spline-Funktionen und Design der Karosserie eines BMW 745i

Eine Spline-Funktion ist eine Funktion, die stückweise auf Teilintervallen definiert ist und mindestens die Eigenschaft erfüllt, dass zwei an der Grenze zweier Intervalle aufeinandertreffende Teilfunktionen denselben Funktionswert haben. Ursprünglich wurden Splines im Schiffsbau eingesetzt, um die Form der Schiffsbeplankung bzw. der einzelnen Straklatten (engl. *splines*), die in Längsrichtung des Rumpfes verlaufen, zu bestimmen. Diese durch Biegelineale bestimmten Splines haben die Eigenschaft, die auf sie wirkenden Kräfte optimal auf die Fixierungspunkte zu verteilen. Daraus entsteht eine Funktion, die sich ideal zwischen die Punkte formt. Heute werden Spline-Funktionen in diversen Anwendungsgebieten eingesetzt. Die Modellierung von Autokarosserien und die Konstruktion von Tragflächen und Turbinen, um nur wenige Beispiele zu nennen, wären ohne Splines kaum denkbar.

Lineare, quadratische und kubische Splines

Man unterscheidet verschiedene Arten von Splines. Die einfachste Art besteht darin, lineare Verbindungen bzw. lineare Funktionen zwischen den gegebenen Punkten, den sogenannten *Stützpunkten*, zu bilden.

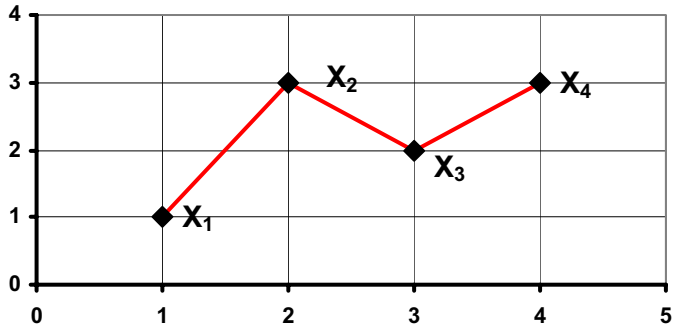


Abb.1

Sind n Stützpunkte $(x_i; y_i)$, $i = 1,2,...,n$, gegeben, so lässt sich im i-ten Intervall $[x_i\ , x_{i+1}]$ die Teilfunktion $sp_i(x) := a_i x + b_i$ natürlich festlegen durch die Bedingungen $sp_i(x_i) = y_i$ und $sp_i(x_{i+1}) = y_{i+1}$ $(i \le n-1)$. Dadurch entsteht allerdings eine Funktion mit Knickstellen, die für viele praktische Anforderungen ungeeignet ist.

Oft vorteilhafter ist die Verwendung *quadratischer Splines*. Dabei wird im i-ten Intervall die quadratische Funktion $sp_i(x) := a_i x^2 + b_i x + c_i$ angesetzt und durch die Bedingungen festgelegt, dass $sp_i(x_i) = y_i$, $sp_i(x_{i+1}) = y_{i+1}$, $sp_i'(x_i) = sp_{i-1}'(x_i)$ und $sp_i'(x_{i+1}) = sp_{i+1}'(x_{i+1})$, wobei $1 < i < n$. Dadurch entsteht eine Funktion, die nicht nur die Minimalbedingung der Stetigkeit erfüllt, sondern überdies eine stetige Ableitung besitzt, eine Eigenschaft, die in vielen Anwendungen erwünscht ist. Allerdings weisen nun zwei in einer Stützstelle aufeinandertreffende Teilfunktionen nicht notwendigerweise dieselbe Krümmung auf; die zweite Ableitung ist noch unstetig. Soll die Funktion etwa eine projektierte Strasse modellieren, die durch die Stützpunkte zu verlaufen hat, so wären quadratische Splines ungenügend, da sich die Beschleunigung in den Stützstellen sprunghaft ändern würde.

Am häufigsten werden daher sogenannte *kubische Splines* verwendet. Dabei wird im i-ten Intervall die kubische Funktion $sp_i(x) := a_i x^3 + b_i x^2 + c_i x + d_i$ angesetzt, und die Koeffizienten werden durch die Forderungen bestimmt, dass eine stetige Funktion durch die Stützstellen entsteht, diese Funktion eine stetige Ableitung besitzt und überdies auch die zweite Ableitung stetig ist.

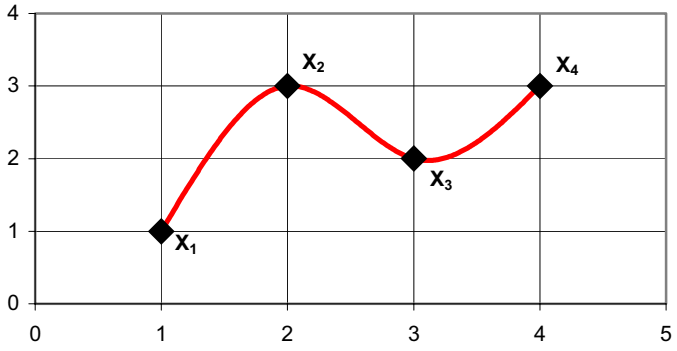


Abb.2

Liegen beispielsweise vier Datenpunkte $(x_1; y_1),...,(x_4; y_4)$ vor, so müssen drei kubische Funktionen mit insgesamt 12 unbekannten Koeffizienten bestimmt werden. Dazu dienen die 6 Gleichungen

$$sp_1(x_1) = y_1$$
$$sp_1(x_2) = y_2$$
$$sp_2(x_2) = y_2$$
$$sp_2(x_3) = y_3$$
$$sp_3(x_3) = y_3$$
$$sp_3(x_4) = y_4$$

die für die Stetigkeit der Funktion sorgen, ferner die beiden Gleichungen

$$sp_1'(x_2) = sp_2'(x_2)$$
$$sp_2'(x_3) = sp_3'(x_3)$$

die eine stetige Ableitung sicherstellen, und die beiden Gleichungen

$$sp_1''(x_2) = sp_2''(x_2)$$
$$sp_2''(x_3) = sp_3''(x_3)$$

die dieselbe Krümmung bei Annäherung an eine Stützstelle von links oder von rechts verlangen. Da nun noch zwei Gleichungen fehlen, fordert man zusätzlich, dass die Funktion an den Enden *linear fortsetzbar* ist, d.h. dass

$$sp_1''(x_1) = 0 \text{ und } sp_3''(x_4) = 0 ,$$

eine vernünftige Forderung, wenn man bedenkt, dass ein Biegelineal, eingespannt in die Stützpunkte, sich genau so verhalten würde.

Es darf nicht verschwiegen werden, dass das Problem, eine Funktion durch n gegeben Datenpunkte zu legen, natürlich auch mit einer polynomischen Interpolation gelöst werden könnte. Der Nachteil einer *einzigen* Polynomfunktion $(n-1)$ -ten Grades für alle n Punkte ist allerdings, dass diese bis zu $n-2$ Extrema haben kann und wesentlich stärker „ausschlägt“ als eine aus Splines aufgebaute Funktion.

Ein Beispiel mit Hilfe von MATLAB

Im Folgenden sollen nun kubische Splines durch vier konkrete Stützpunkte gelegt werden. Als Beispiel wählen wir die Punkte A=(1;1), B=(2;3), C=(3;2), D=(4;3) (vgl. Abb.2). Da $sp_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$ ist, folgt: $sp_i'(x) = 3a_i x^2 + 2b_i x + c_i$ und $sp_i''(x) = 6a_i x + 2b_i$, und es entstehen, wenn wir die oben verwendete Reihenfolge übernehmen, die folgenden 12 Gleichungen:

(I) $a_1 + b_1 + c_1 + d_1 = 1$
(II) $8a_1 + 4b_1 + 2c_1 + d_1 = 3$
(III) $8a_2 + 4b_2 + 2c_2 + d_2 = 3$
(IV) $27a_2 + 9b_2 + 3c_2 + d_2 = 2$
(V) $27a_3 + 9b_3 + 3c_3 + d_3 = 2$
(VI) $64a_3 + 16b_3 + 4c_3 + d_3 = 3$
(VII) $12a_1 + 4b_1 + c_1 = 12a_2 + 4b_2 + c_2$
(VIII) $27a_2 + 6b_2 + c_2 = 27a_3 + 6b_3 + c_3$
(IX) $12a_1 + 2b_1 = 12a_2 + 2b_2$
(X) $18a_2 + 2b_2 = 18a_3 + 2b_3$
(XI) $6a_1 + 2b_1 = 0$
(XII) $24a_3 + 2b_3 = 0$

Dies führt zu folgender 12 x 12 Matrix:

	a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	c_3	d_1	d_2	d_3	
1 =	1	0	0	1	0	0	1	0	0	1	0	0	I
3 =	8	0	0	4	0	0	2	0	0	1	0	0	II
3	0	8	0	0	4	0	0	2	0	0	1	0	III
2 =	0	27	0	0	9	0	0	3	0	0	1	0	IV
2 =	0	0	27	0	0	9	0	0	3	0	0	1	V

3 =	0	0	64	0	0	16	0	0	4	0	0	1	VI
0 =	12	-12	0	4	-4	0	1	-1	0	0	0	0	VII
0 =	0	27	-27	0	6	-6	0	1	-1	0	0	0	VIII
0 =	12	-12	0	2	-2	0	0	0	0	0	0	0	IX
0 =	0	18	-18	0	2	-2	0	0	0	0	0	0	X
0 =	6	0	0	2	0	0	0	0	0	0	0	0	XI
0 =	0	0	24	0	0	-2	0	0	0	0	0	0	XII

Nennen wir diese Matrix A, so lässt sich das lineare Gleichungssystem $A \cdot \vec{x} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ 2 \\ 2 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ leicht in MATLAB auflösen

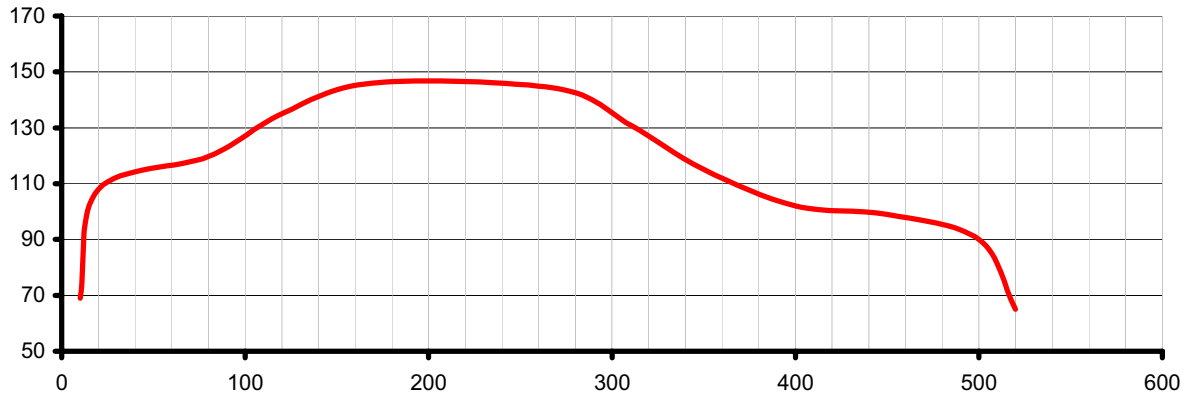
und liefert schliesslich die drei kubischen Splines

$$sp_1(x) = -0.93x^3 + 2.8x^2 + 0.13x - 1$$
$$sp_2(x) = 1.67x^3 - 12.8x^2 + 31.33x - 21.8$$
$$sp_3(x) = -0.73x^3 + 8.8x^2 - 33.47x + 43$$

deren graphische Darstellung in Abb.2 gegeben ist.

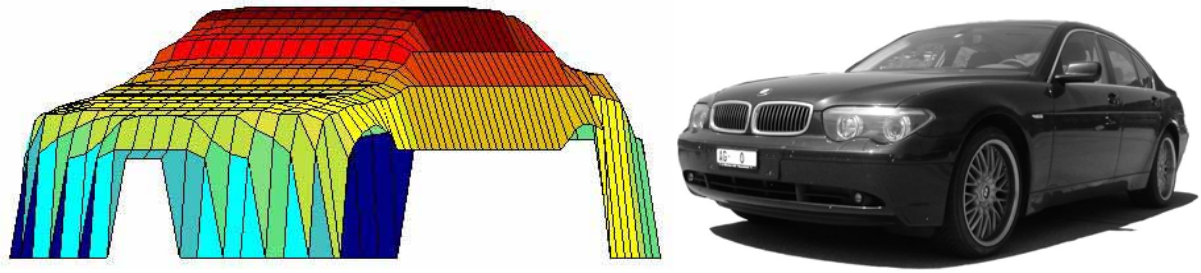
Design der Karosserie eines BMW 745i

Als etwas umfangreichere Anwendung der Splines, in der MATLAB (oder eine andere starke Mathematiksoftware) unverzichtbar ist, haben wir uns vorgenommen, die Karosserie eines Autos mathematisch zu modellieren. Wir haben dazu ein Gitternetz mit einem Raster von 10 cm über das gesamte Auto gelegt und 52x19=988 Stützpunkte ausgemessen, wobei aus Symmetriegründen natürlich nur die Hälfte, also etwa 500 Messdaten, tatsächlich zu erheben waren. Schneidet man (im übertragenen Sinne des Wortes) den BMW entlang seiner Längsrichtung, so fallen 52 Stützpunkte und ein analoges Problem wie im obigen Beispiel an. MATLAB löst das System, liefert die einzelnen Splines und stellt den Längsschnitt graphisch dar:



Schliesslich nahmen wir uns vor, ein dreidimensionales Modell der ganzen Karosserie zu erhalten. Die insgesamt 52x19=988 Stützpunkte sollten sowohl in Längs-, wie auch in Querrichtung durch kubische Splines verbunden werden, und wir strebten eine dreidimensionale Darstellung an. Um vorerst nur die Messwerte zu visualisieren, gaben wir sie in MATLAB ein:

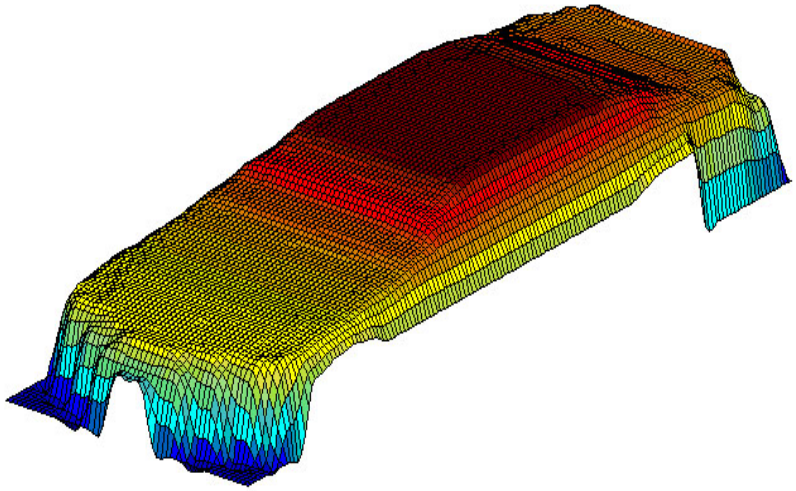
```
x=0:10:510;
y=0:10:180;
z=[0 0 0 0 …;0 69 70 82 …;0 72 82 100; …];
surf(x,y,z)
```



Um nun sowohl in x- als auch in y-Richtung Splines zwischen je zwei benachbarten Datenpunkten zu erhalten, erweitern wir das Programm wie folgt:

```
xi=linspace(0,510,153);
yi=linspace(0,180,54);
[xxi,yyi]=meshgrid(xi,yi);
zzi=interp2(x,y,zzi,xxi,yyi,'cubic');
surf(xxi,yyi,zzi)
```

Das Datennetz besteht nun aus rund 8900 Datenpunkten, und MATLAB liefert ein aus Splines aufgebautes graphisches Modell der Karosserie.



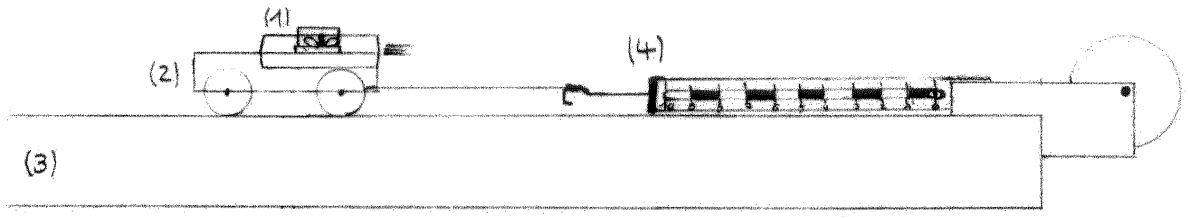
III. Raketenstart – Modell und Wirklichkeit

Bei dieser Anwendung war es unser Bestreben, eine kleine Rakete zu bauen und ihren realen Flug mit den Werten zu vergleichen, die ein auf theoretischen Überlegungen gegründeter Algorithmus liefert. Die Rakete stellten wir mit Hilfe eines Bausatzes her (vgl. Abb.); einige Daten der Rakete und des Treibsatzes können dem untenstehenden Kasten entnommen werden.

Daten Rakete:	Daten Treibsatz:
Länge: 40 cm	Typ: B4-2 BAM Nr. PT1-0128
Durchmesser: 19 mm	Länge: 7 cm
Gewicht ohne Treibsatz: 35 g	Durchmesser: 18 mm
Kein Fallschirm	Gewicht: 20 g
	Angegebene Schubkraft: 8 N

Bestimmung von Schubkraft, Masse, Luftwiderstand und Beschleunigung

Soll ein mathematisches Modell der realen Flugbahn bestimmt werden, so sind dringend Schubkraft und Brenndauer des Treibsatzes zu untersuchen. Dazu bauten wir die abgebildete Versuchsanordnung, in der der Treibsatz auf einem auf einer Schiene geführten (und beschwerten) Wagen befestigt war. Eine Federwaage diente der Bestimmung der auftretenden Schubkraft. Der ganze Versuch wurde gefilmt, damit nachher die Schubkraft F_S zu möglichst vielen Zeitpunkten möglichst genau abgelesen werden konnte.



Die Brenndauer betrug 1.68 s. Durch Interpolation berechneten wir die Schubkraft

$$F_S(t) \approx \begin{cases} -16.64t^4 + 60.70t^3 - 78.03t^2 + 39.39t, \text{ falls } t \le 1.68\text{s} \\ 0, \text{ falls } t > 1.68\text{s} \end{cases}$$

Nun konnten wir uns daran machen, einen Algorithmus zu entwerfen, der die Flugbahn näherungsweise richtig wiedergeben sollte. Besonders reizvoll war, dass dabei sowohl unser Taschenrechner TI-92, als auch einiges von dem, was wir gerade in der Analytischen Geometrie gelernt hatten, zum Einsatz kam. Zunächst ging es darum, ein paar physikalische Beiträge zusammenzustellen:

- Wir nahmen an, dass die Raketenmasse $m(t)$ von anfangs 0.055 kg (Rakete und Treibsatz) während einer Brenndauer von 1.68 s auf 0.045 kg (Rakete und ausgebrannter Treibsatz) linear abnimmt, so dass also

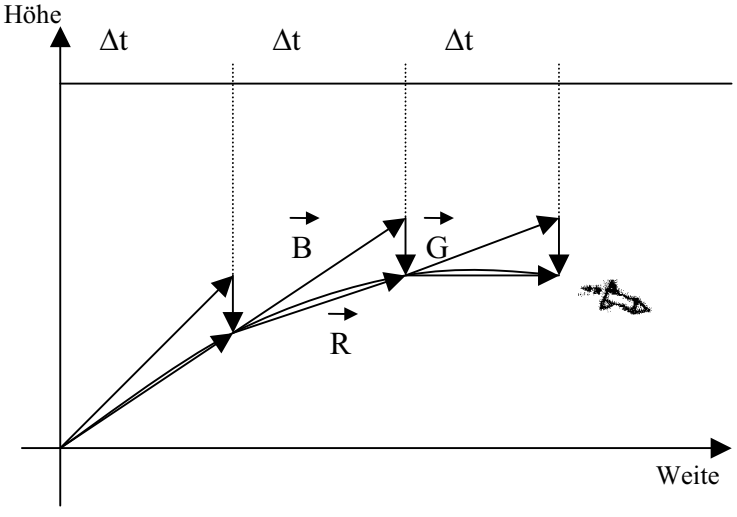
$$m(t) = \begin{cases} 0.055 - \dfrac{0.055 - 0.045}{1.68} t, \text{ falls } t \le 1.68\text{s} \\ 0.045, \text{ falls } t > 1.68\text{s} \end{cases}$$

- Für den Luftwiderstand F_L setzten wir $F_L(v) = c \cdot v^2$ an und bestimmten die Konstante mit Hilfe der Geometrie der Rakete (und der Formelsammlung) zu $c = 0.001325$, so dass also schliesslich $F_L(v) = 0.001325v^2$.

- Die Raketenbeschleunigung ist somit $a_R(t) = \dfrac{F_S(t) - F_L(t)}{m(t)}$.

Die Idee

Nun zerstückelten wir die Flugbahn in Intervalle der Länge $\Delta t = 0.1\text{s}$ und überlegten uns, dass sich in jedem einzelnen Intervall der Streckenvektor $\vec{R}$ der Rakete zusammensetzt aus einem allein aus der Raketenbeschleunigung sich ergebenden Vektor $\vec{B}$ und einem aus der Erdanziehung sich ergebenden Vektor $\vec{G}$ (vgl. Abb.).



Dabei ist $\vec{G} = \begin{pmatrix} 0 \\ -\frac{g}{2}(\Delta t)^2 \end{pmatrix}$ und $\vec{B} = \left(v(t) \cdot \Delta t + \frac{a_R}{2} \cdot (\Delta t)^2 \right) \cdot \vec{O}$, wobei wir unter $\vec{O}$ einen Orientierungsvektor der Länge 1 verstehen, der für die richtige Richtung sorgt; und natürlich kann für $\vec{O}$ in jedem Intervall der auf Länge 1 gestauchte Vektor $\vec{R}$ des vorangehenden Intervalls gewählt werden, und im ersten Zeitintervall muss $\vec{O} = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$ sein, wenn α der Abschusswinkel der Rakete ist. Damit war unser Flugbahnalgorithmus in Griffnähe:

1. Setze $t = 0,\ v(0) = 0,\ \Delta t = 0.1,\ \vec{O} = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$.
2. Berechne in einer Schleife die einzelnen Streckenvektoren $\vec{R}$ der Zeitintervalle durch $\vec{R} = \vec{B} + \vec{G}$, wobei $\vec{B}$ und $\vec{G}$ wie oben definiert sind. Die Geschwindigkeit wird in jedem Intervall um $a_R(t) \cdot \Delta t$ erhöht, und $\vec{R}$ muss am Ende jedes Schleifendurchganges auf Länge 1 normiert werden, um als Richtungsvektor für das nächste Intervall zur Verfügung zu stehen.
3. Durch Ausdruck der einzelnen Vektoren $\vec{R}$ entsteht ein theoretisches Modell der Flugbahn.

Der Flug im Taschenrechner

Im Taschenrechner TI-92 programmiert nimmt der Algorithmus in seiner einfachsten Form die folgende Gestalt an:

```
Rakete()
Prgm
Local a,v,t.dt,ge,o,fs(t),m(t),ar,b,g,r
Clrdraw
Request "Abschusswinkel=",a
Expr(a)→a
:
0→v : 0→t : 0.1→dt : 9.81→ge : [cos(a);sin(a)]→o
:
Loop
  If t<1.68 Then
    -16.64*t^4+60.7*t^3—78.03*t^2+39.39*t→fs(t)
  Else
    0→fs(t)
  Endif
  If t<1.68 Then
    0.055-(0.01/1.68)*t→m(t)
  Else
    0.045→m(t)
  Endif
  (fs(t)-0.001325*v^2)/m(t)→ar
  (v*dt+0.5*ar*dt^2)*o→b
  If t=0 Then
    [0;0]→g
  Else
    [0;-(ge/2)*dt^2]→g
  Endif
  b+g→r
  If t>0 Then
    UnitV(r)→o
  Endif
  v+ar*dt-->v
  t*dt→t
  Disp r
```

```
  Pause
Endloop
:
Endprgm
```

Nun liessen wir unsere Rakete fliegen und stellten eine erstaunlich gute Übereinstimmung des realen Fluges mit dem theoretisch vorhergesagten Flug fest: Mit einem Abschusswinkel von 90° stieg die Rakete ca. 133 m hoch, während unser Algorithmus eine Maximalhöhe von 122 m vorhersagte. Bei einem Abschusswinkel von 45° massen wir eine Flugweite von 96 m, während der Taschenrechner eine Weite von 109 m angab. Auch die Flugbahn stimmte überraschend gut mit der Kurve überein, die wir mit Taschenrechnerhilfe zeichnen liessen.