Knoten aus der vierten Dimension

Burkhard Wald, Universität GH Essen, Hochschulrechenzentrum

Wir wollen aufzeigen, wie man ausgehend von einer Maple-Berechnung, eine 3D-Grafik in verschiedenen Formaten erhalten kann. Zunächst mal hat man die dreidimensionale Grafik innerhalb eines Maple-Workscheets. Hier kann man mit der Maus in die Grafik hineinklicken um das dargestellte Objekt zu drehen.

Exportiert man das Worksheet als HTML-Dokument um es als Web-Seite anzubieten, so ist die 3D-Graphik zu einem statischen 2D-Bild geworden. Wir wollen hier zwei Möglichkeiten vorstellen, das 3D-Objekt wieder rotierbar zu machen. Die erste Möglichkeit ist das 3D-Format VRML, für das Maple eine Exportschnittstelle besitzt. Die andere Möglichkeit ist ein kleines Java-Applet, das zum Beispiel-Umfang einer Java-Entwicklungsumgebung gehört. So läßt sich ein mit der Maus bewegbares Drahtgitter direkt in eine Web-Seite einbinden. Als Drittes wollen wir dann den Raytracer Povray benutzen, um ein photorealistisches Bild zu erzeugen.

Wir wollen diesem Artikel aber auch einen mathematischen Inhalt geben. Das Worksheet entstand aus dem Versuch heraus, einmal etwas nachzurechnen, was in meiner mathematischen Vorstellungswelt immer noch mit wabernden Nebeln umgeben war. Knoten, also in sich verschlungene Kurven, sind Projektionen von vierdimensionalen Objekten, die durch relativ einfache Gleichungen entstehen. So sollen aus der Gleichung **x^2=y^2** zwei ineinander hängende Kreise werden - heißt es. Nun, wir zeigen, wie es geht.

```
> restart;
```

Aufstellen der Gleichungen

Wir beginnen mit der Gleichung

```
> Eq1:=x^2=y^2;
```

$$Eq1 := x^2 = y^2$$

wollen das aber innerhalb der complexen Zahlen betrachten.

```
> subs(x=a+b*I,y=c+d*I,Eq1);
```

$$(a + I\,b)^2 = (c + I\,d)^2$$

```
> Eq2:=evalc(%);
```

$$Eq2 := a^2 + 2\,I\,a\,b - b^2 = c^2 + 2\,I\,c\,d - d^2$$

Separieren wir hier den imaginären Anteil.

```
> tmp:=map(z->select(has,z,I),Eq2);
```

$$tmp := 2\,I\,a\,b = 2\,I\,c\,d$$

Aus dem reellen ergibt sich

```
> Eq3:=Eq2-tmp;
```

$$Eq3 := a^2 - b^2 = c^2 - d^2$$

und aus dem imaginären Anteil

> **Eq4:=tmp/(2*I);**

$$Eq4 := a\,b = c\,d$$

Eq3,Eq4 sind jetzt die Gleichungen in den Variablen **a,b,c,d** von denen wir ausgehen. Somit haben wir irgend ein Gebilde im 4-dimensionalen reelen Raum. Dort wollen wir das mit der Einheitskugel schneiden. Dadurch kommt die folgende Gleichung **Eq5** hinzu.

> **Eq5:=a^2+b^2+c^2+d^2=1;**

$$Eq5 := a^2 + b^2 + c^2 + d^2 = 1$$

Auswerten der Gleichungen

Wir wollen die drei Gleichungen jetzt zusammen auswerten, um uns die Situation vor Augen zu führen. Statt Eq4 arbeiten wir aber besser mit dem Quadrat der Gleichung.

> **map(x->x^2,Eq4);**

$$a^2\,b^2 = c^2\,d^2$$

Das setzen wir jetzt alles in einen Aufruf von **solve** ein.

> **res:=solve({Eq3,%,Eq5},{b^2,c^2,d^2});**

$$res := \{c^2 = a^2,\ d^2 = -a^2 + \frac{1}{2},\ b^2 = -a^2 + \frac{1}{2}\}$$

Daraus ergeben sich die folgenden 3 Gleichungen.

> **E1:=res[3]+(a^2=a^2);**

$$E1 := b^2 + a^2 = \frac{1}{2}$$

> **E2:=res[1];**

$$E2 := c^2 = a^2$$

> **E3:=res[2]-res[3]+(b^2=b^2);**

$$E3 := d^2 = b^2$$

Somit wird zwischen a und b ein Kreis beschrieben und es ist c=a oder c=-a sowie d=b oder d=-b. Aus

> **Eq4;**

$$a\,b = c\,d$$

folgt aber, dass entweder c=a und d=b, oder c=-a und d=-b ist.

Somit sind die 4-dimensionalene Tupel **[a,b,a,b]** und **[a,b,-a,-b]** mit **a^2+b^2=1/2** genau die gesuchten Lösungen. Das sind zwei Kreise, die auf zwei Arten diagonal in den 4-dimensionalen Raum eingebettet sind. Wir benötigen eine Parametrisierung der Kreise.

> **K1:=[r*sin(alpha), r*cos(alpha), r*sin(alpha), r*cos(alpha)];**

$$K1 := [r \sin(\alpha), r \cos(\alpha), r \sin(\alpha), r \cos(\alpha)]$$

> **K2:=[r*sin(alpha), r*cos(alpha), -r*sin(alpha), -r*cos(alpha)];**

$$K2 := [r \sin(\alpha), r \cos(\alpha), -r \sin(\alpha), -r \cos(\alpha)]$$

mit

> **r:=1/sqrt(2);**

$$r := \frac{1}{2}\sqrt{2}$$

Die Projektion in die wirkliche Welt

Jetzt geht es darum, das Gebilde in einen 3-dimensionalen Raum zu projizieren. Dazu begeben wir uns zu einem Punkt außerhalb unseres Objektes, das ja auf jeden Fall auf der Einheitskugel liegt. Sagen wir

> **v:=[2,0,0,0];**

$$v := [2, 0, 0, 0]$$

Von dort aus projizieren wir alles auf die senkrecht dazu stehende Hyperebene

> **[1,b,c,d];**

$$[1, b, c, d]$$

Für einen Punkt w bilden wir dazu die Differenz w-v und normieren die erste Koordinate auf 1 .

> **P:=w->(w-v)[2..4]/(w-v)[1];**

$$P := w \rightarrow \frac{(w - v)_{2\,..\,4}}{(w - v)_1}$$

> **P([a,b,c,d]);**

$$\frac{[b, c, d]}{-2 + a}$$

Jetzt projizieren wir die beiden Kreise.

> **PK1:=P(K1);**

$$PK1 := \frac{\left[\frac{1}{2}\sqrt{2}\,\cos(\alpha), \frac{1}{2}\sqrt{2}\,\sin(\alpha), \frac{1}{2}\sqrt{2}\,\cos(\alpha)\right]}{-2 + \frac{1}{2}\sqrt{2}\,\sin(\alpha)}$$

> **PK1:=expand(PK1);**

$$PK1 := \left[\frac{1}{2} \frac{\sqrt{2}\cos(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)}, \frac{1}{2} \frac{\sqrt{2}\sin(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)}, \frac{1}{2} \frac{\sqrt{2}\cos(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)} \right]$$

> **PK2:=P(K2);**

$$PK2 := \frac{\left[\frac{1}{2}\sqrt{2}\cos(\alpha), -\frac{1}{2}\sqrt{2}\sin(\alpha), -\frac{1}{2}\sqrt{2}\cos(\alpha) \right]}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)}$$

> **PK2:=expand(PK2);**

$$PK2 := \left[\frac{1}{2} \frac{\sqrt{2}\cos(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)}, -\frac{1}{2} \frac{\sqrt{2}\sin(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)}, -\frac{1}{2} \frac{\sqrt{2}\cos(\alpha)}{-2+\frac{1}{2}\sqrt{2}\sin(\alpha)} \right]$$

Das Resultat mit Maple gezeichnet

Mit der Funktion **tubeplot** geben wir diesen beiden Kurven im 3-dimensionalen etwas Volumen (aus Kreise werden Ringe).

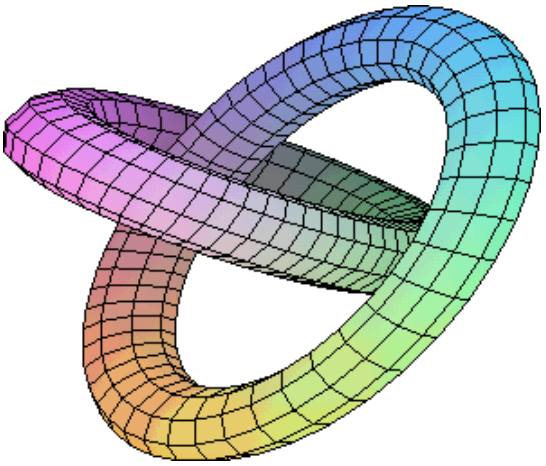
> **with(plots):**

> **R1:=tubeplot(PK1,alpha=0..2*Pi,radius=0.1):**

> **R2:=tubeplot(PK2,alpha=0..2*Pi,radius=0.1):**

So, jetzt schauen wir uns einmal beide Ringe zusammen an. Es ist verblüffend - wie aus dem Hut gezaubert.

> **display([R1,R2]);**



> **R:=%:**

Virtuelle Realität

Wir wollen daraus nun ein VRML-Modell erzeugen. VRML bedeutet "Virtual Reality Modelling Language" und ist eine Sprache, mit der man 3-dimensionale Szenen und Objekte beschreiben kann. Man benötigt natürlich einen speziellen Browser, um sich solche VRML-Dateien anschauen zu können. Für Window-PCs bietet sich der Cosmo-Player der Firma SGI an. Er ist ein Plug-In für den Netscape-Navigator oder den Internet-Explorer und kann frei heruntergeladen werden. Wer mehr über VRML wissen möchte, sei auf einen Artikel in einer früheren HeRZ-Blatt-Ausgabe verwiesen. Die Erzeugung der VRML-Datei aus Maple heraus ist denkbar einfach.

> **with(plottools,vrml);**

$$[\,vrml\,]$$

> **vrml(R,`knoten.wrl`);**

Sie können jetzt einen HTTP-Link zu dieser Datei einfügen. Das macht man mit dem Menü "Insert-Hyperlink". Dieser Link bleibt natürlich bestehen, wenn Sie das Worksheet über "File-Export As" nach HTML konvertieren (also in eine Web-Seite verwandeln).

Hier ist der Link zur VRML-Datei

Und so sieht das Netscape-Fenster mit dem Cosmoplayer-PugIn aus.

Draht-Modell durch Java-Applet

In der Einleitung haben wir noch eine zweite Variante versprochen. Will man Web-Seiten mit einer Funktionalität ausstatten, so sind die Programmiersprachen JAVA und JAVASCRIPT geeignet. Die heute gängigen WWW-Browser sind in der Lage Java- und Javascript-Programme im Browser ablaufen zulassen. Das bietet die Möglichkeit, dass man einen 3D-Viewer programmiert, und diesen einfach an eine Web-Seite anhängt. Dafür erübrigt sich das Installieren eines speziellen Plug-Ins. Solch ein Java-3D-Viewer ist ein verbreitetes Java-Beispiel, das auf viele Jave-Sites zu finden ist. Es zeichnet ein einfaches Drahtmodell eines Objektes und bietet die Möglichkeit das Objekt mit der Maus zu drehen. "Hidden Lines" werden nicht berechnet, aber weiter hinten liegende Linien werden heller dargestellt als vorne liegende Linien. Dadurch entsteht eine ganz brauchbare Objektdarstellung. In dem Programm mußte ich aber für meine Zwecke eine kleine Änderung vornehmen. Es war auf das Zeichnen geschlossener Polygone ausgerichtet und zog deshalb immer noch eine zusätzliche Linie vom letzten Punkt eines Polygons zum ersten Punkt. Das ließ sich aber leicht herausnehmen. Ganz so problemlos ist der Umgang mit Java aber nicht. Im Gegensatz zu Javascript benötigt man für Java einen Übersetzer, der ein Java-Programm in ein binäres Format übersetzt. Nur dieses übersetzte Format kann dann von einer abstrakten Java-Maschine abgearbeitet werden. Nun ja, zumindest eine Hemmschwelle: man muß sich eine Java-Entwicklungsumgebung besorgen, sie installieren und lernen wie man damit umgeht. Die übersetzten Java-Programme, die dann im WWW-Browser ablaufen, heißen übrigens "Applets" und haben die Dateiendung .class .

Das Java-Applet muß natürlich auch irgend woher eine Beschreibung des darzustellenden Objektes lesen. Das Datei-Format hierfür ist sehr einfach. Zuerst stehen dort die x-y-z-Werte der Punkte und dann die Polygone als Liste von Nummern der zu verbindenden Punkte. Kennt man nun auch die Struktur mit der Maple 3D-Plots verwaltet, so läßt sich leicht eine Maple-Prozedur schreiben, die das gewünschte Format in eine Datei schreibt. Wir wollen hier auf Details nicht eingehen, und lesen die Prozedur aus einer Datei.

```
> read(`/u/hrz030/maple/lib/plot2java`);
```

```
> plot2java(R);
```

Die Prozedur hat automatisch einen Dateinamen vergeben und schreibt genau den HTML-Befehl, den man zur Einbindung in eine Web-Seite benötigt, in das Worksheet. Leider ist der HTML-Exporter von Maple so intelligent, dass er diesen String auch genauso im HTML-Dokument wiedergibt.

```
_HTML:<applet code=ThreeD.class width=500 height=500><param name=model value=Maple_temp.1.obj></applet>
```

Es wird also verhindert, dass die HTML-Tags interpretiert werden. Daher rücke ich dem von Maple geschriebenen

HTML-Dokument noch mal mit einem Perl-Script zu Leibe, das die Maskierung der Symbole < und > wieder herausnimmt. Dann ist an Stelle des Prozeduren-Outputs das Applet mit dem 3D-Objekt zu sehen.

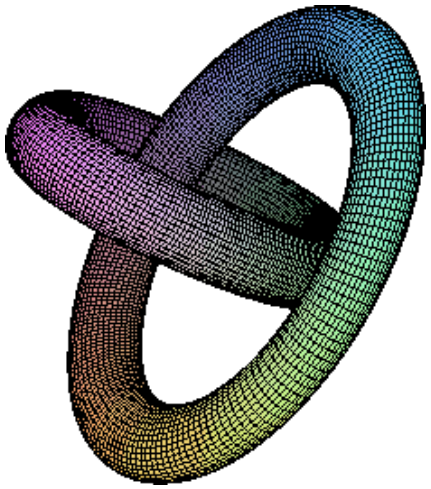
Ein photoreales Bild

Als nächstes wollen wir ein photorealistisches Bild erzeugen. Hierzu gibt es eine kleines Programm eines norwegischen Maple-Nutzers, das eine Maple-Plotstruktur in das Eingabeformat des Raytracers POVRAY bringt. Wir benötigen den Maple-Plot aber mit einer feineren Gitterstruktur.

```
> R1:=tubeplot(PK1,alpha=0..2*Pi,radius=0.1,grid=[200,40]):
```

```
> R2:=tubeplot(PK2,alpha=0..2*Pi,radius=0.1,grid=[200,40]):
```

```
> display({R1,R2});
```



Das schreiben wir nun in eine Datei.

```
> R:=%:
```

```
> writeto(`/temp/knoten`);
```

```
> lprint(R);
```

```
> writeto(terminal);
```

```
> back;
```

back

Mit dem Programm povray und der Schnittstelle m2p (maple-to-povray) wird nun ein Pixelimage erzeugt. Für beides geben wir unten noch Download-Hinweise.

```
> system(`m2p -sW /temp/knoten`);
```

0

Mit diesem Befehl hat man eine Datei /temp/knoten.pov erzeugt. Darin steht die Beschreibung der 3D-Scene wie Povray sie lesen kann. Sie ist in unserem Fall über 10 MB groß.

> **system(`povray +I/temp/knoten.pov +O/temp/knoten.tga +H400 +W400 +FT +L/temp/hrz030/m2p_ptrace`);**

13824

Die Output-Datei ist im Targa-Format. Daraus müssen wir nun noch eine GIG-, JPEG- oder PNG-Datei machen, damit man sie z.B. in eine Web-Seite einbinden kann. Sehr einfach geht das mitdem Befehl convert aus dem Graphik-Programm ImageMagick.

> **system(`convert /temp/knoten.tga knoten.gif`);**

0

In Bezug auf die letzten drei Aktionen muß erwähnt werden, dass das Maple-Woorksheet auf einem Unix-Rechner ausgeführt wurde. Das Programm m2p liegt als C-Quelle vor. Es ließ sich zwar auch auf einem NT-Rechner problemlos compilieren, doch funktionierte es nur bei sehr kleinen Beispielen. (Das Programm ist aber nicht so komplex, dass man nicht versuchen könnte das Problem zu klären.) Das Povray-Paket gehört zu der Platte /sw die man sich vom HRZ mounten kann (wir haben darüber berichtet). Auf der SP2 steht es deshalb allen Nutzern zur Verfügung. Das ist allerdings nur eine Kommandozeilen-Version von Povray, was den Nachteil hat, dass man sich einmal die verschiedenen Komandozeilen-Parameter in der Dokumentation anschauen muß. Es hat aber den Vorteil, dass wir das Kommando in Maple fest einbinden können. Für PCs gibt es eine sehr schöne Windows-Klick-Version von Povray. In diesem Fall legt man das Maple-Fenster nach dem Erzeugen der Povray-Datei bei Seite, startet Povray für Windows, Datei laden, u.s.w. Die TGA-Datei kann man unter Windows z.B. mit Coral-Draw konvertieren. Die neueren Versionen von Povray können aber auch von vornherein PNG-Dateien erzeugen. Damit geht man auch lizenzrechrechtlichen Problemen mit dem GIF-Format aus dem Weg.

Überflüssige Dateien können wir wieder löschen.

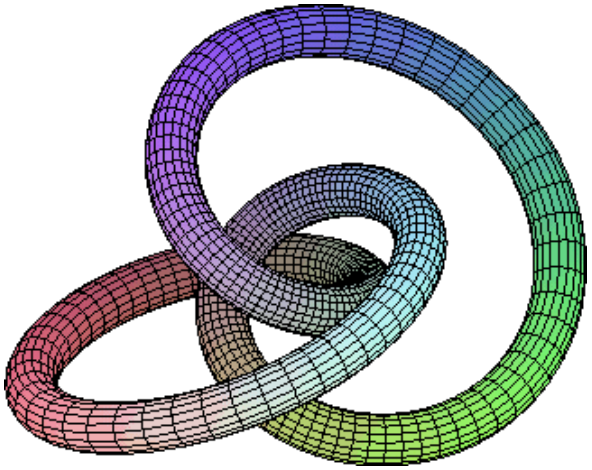
> **system(`rm /temp/knoten*`);**

0

Noch mehr Knoten

Es es gibt ein Maple-Paket mit dem Namen algcurves, das zum Standartumfang gehört und eine Funktion plot_knot enthält. Damit lassen sich solche Kurven in einer Black-Box-Anwendung erzeugen. Das Titelbild zu dieser HeRZ-Blatt-Ausgabe wurde so erzeugt. Zunächst wird das Bild als Maple-Graphik geplottet.

```
> algcurves[plot_knot](y^3-x^2,x,y,epsilon=0.8,radius=0.18,grid=[200,20]);
```



Wie wir es oben beschrieben haben wurde das danach mit mit m2p und Povray weiterverarbeitet. Hier sollte es schon etwas schwieriger sein, sich das aus der Gleichung y^3-x^2=0 resultierende Objekt im 4-dimensionalem Raum vorzustellen. Die schwierigste Aufgabe, die die Prozedure plot_knot leisten muß, ist es aber eine Parametriesierung der Kurve anzugeben. In dem von uns oben gerechneten Beispiel hatten wir es mit den zwei Kreisen doch sehr leicht.

Verweise

Informationen zu Maple:
http://www.uni-essen.de/hrz/mathe/maple

Online-Version dieses Artikels mit anklickbaren Hyperlinks und andere Worksheets:
http://www.uni-essen.de/hrz/mathe/maple/worksheets

Ebenfalls zu finden im Archiv für elektronische Dokumente MILESS:
http://miless.uni-essen.de

Informationen zu VRML und Cosmo-Player:
http://www.uni-essen.de/hrz/vrml

Die erwähnte Java-Entwicklungsumgebung findet man an der Universität Tübingen in einem Archiv mit dem Titel " Kostenlose und geprüfte NT-Software":
http://ntfreeware.zdv.uni-tuebingen.de

Die benutzte Maple-Prozedure zum Erzeugen des Java-Drahtgitters erhalten Sie auf Anfrage.
mailto:wald@hrz.uni-essen.de

Das Programm m2p ist von Björn Vaggen Kanestabo vom Mathematischen Institut der Universität Oslo.
http://www.math.uio.no/m2p/

Die Home-Page von Povray:
http://www.povray.org/

Informationen zum Thema Knoten (Bilder, Animationen und weitere Links) findet man auf der Home-Page des Mathematikers Brian Sandersons
http://www.maths.warwick.ac.uk/~bjs