Arbeitsblatt

Gewoehnliche Differentialgleichungen mit Maple

PD Dr. F. Hettlich

```
> restart;
```

1 Einleitung

Da Sie in diesem Worksheet keine eigenen Eintraege machen koennen, empfehle ich zunaechst ein weiteres Fenster zu oeffnen, in dem Sie Befehle austesten und die Aufgaben bearbeiten koennen. Dabei sollten Sie selbstverstaendlich die hier vorgeschlagenen Loesungen zu den Aufgaben erst nach eigenen Versuchen betrachten.

Die angegebenen Befehle (rot) im Worksheet koennen Sie durch "enter" aktivieren und sich das Ergebnis ansehen. Darueberhinaus sind "Links" (gruen) zu den Maple-Beschreibungen eingebaut, die Sie durch anklicken mit der linken Maustaste gezeigt bekommen. Wenn Sie ausversehen einen Befehl doppelt ausgefuehrt haben und dadurch Variablen veraendert wurden, muessen Sie zunaechst mit restart (s. oben) neu anfangen und danach den Abschnitt nochmal von vorn ausfuehren.

Nun wuensche ich Ihnen eine spannende Entdeckungsreise in die Welt der Differentialgleichungen.

1.1 Grundlagen (Funktionen in Maple)

1.2 Differentialgleichungen in Maple

Gewoehnliche Differentialgleichungen (DGL) sind allgemein durch

$$F(\ y(x)\ ,\ Dy(x)\ ,\ldots,\ D^n y(x)\ ,\ x\ ) = 0$$

gegeben. Dabei wird durch den Grad n der hoechsten auftretenden Ableitung die Ordnung der DGL festgelegt. Solche Ausdruecke koennen direkt in der Maple Syntax eingeben werden. So laesst sich z.B. die DGL

$$(x^2+1\ )\ y\ '(x\ ) + 2\,x\ \ y(x) = 0$$

durch den Befehl

```
> dgl:= (x^2+1)*D(y)(x) + 2*x*y(x) = 0;
```

angeben. Eine Vielzahl von Techniken zur analytischen Loesung einer DGL steht nun mit dem entscheidenden Befehl dsolve zur Verfuegung.

```
> dsolve(dgl);
```

Da wir keinen Anfangswert zu der DGL erster Ordnung festgelegt haben, wird die allgemeine Loesung mit einer Konstante "c_1" angeben. Eine Anfangsbedingung kann aber auch beruecksichtigt werden.

```
> abed:= y(0) = 1;
> lsg:=dsolve(dgl,abed,y(x));
```

Beachten Sie die Syntax. Die DGL zusammen mit einer moeglichen Anfangsbedingung ist das erste Argument in Form einer Menge. Bei dieser Variante ist es notwendig, auch das zweite Argument, die gesuchte Funktion, mit anzugeben.

Das Resultat von dsolve ist ein Ausdruck(!) und kann mit den allgemeinen Befehlen weiter bearbeitet werden. Zum Beispiel koennen wir uns den Graphen der Loesung ansehen.

```
> plot(rhs(lsg),x);
```

(rhs liefert uns dabei die rechte Seite der Gleichung in dem Ausdruck *lsg* .)

Damit haben wir schon den wichtigsten Befehl kennengelernt. Eine Vielzahl speziellerer Befehle zur Behandlung von Differentialgleichungen, Differentialgleichungssytemen und partiellen Differentialgleichungen ist in einem Programmpaket DEtools zusammengestellt. Die Befehle lassen sich aber erst nutzen, wenn dieses Paket zusaetzlich geladen wird.

```
> with(DEtools);
```

Es kann natuerlich nur ein kleiner Teil der aufgelisteten Befehle angesprochen werden. Viele sind fuer uns auch nicht weiter von Interesse, da nur gewoehnliche Differentialgleichungen betrachtet werden.

Hier in der Einleitung seien nur zwei Komandos herausgegriffen. Beim Befehl *dsolve* wird der DGL zunaechst ein Typ zugeordnet, der ueber die anzuwendende Loesungsstrategie entscheidet. Diese Zuordnung erhalten wir mit dem Befehl odeadvisor .

```
> odeadvisor(dgl);
```

Wenn Sie sich die Hilfedatei dazu ansehen, bekommen Sie einen guten Eindruck von den Moeglichkeiten, die Maple bereitstellt. Ein weiterer nuetzlicher Befehl ist odetest, mit dem verifiziert werden kann, ob ein Ausdruck eine DGL loest.

```
> odetest(y(x)=4/(1+x^2),dgl); odetest(y(x)=x^2,dgl);
```

Im ersten Fall erhalten wir die Ausgabe 0. Also ist die DGL erfuellt. Im zweiten Fall hingegen erhalten wir den von Null verschiedenen Rest nach einsetzen in die DGL. In unseren Beispielen laesst sich eine vermeintliche Loesung auch mit dem Befehl subs testen.

```
> f:=x->4/(1+x^2); subs(y=f,dgl); simplify(%);
```

Mit odetest lassen sich auch bequem implizit gegebene Loesungen und Loesungen zu Differentialgleichungssystemen ueberpruefen. Im folgenden wird auf das Verifizieren von Loesungen, die durch *solve* oder *dsolve* bestimmt wurden,

verzichtet. Im allgemeinen sind solche Tests aber angebracht - Fehler verstecken sich gerne!

Aufgabe: Bestimmen Sie die Loesungen zu folgenden Anfangswertproblemen. Betrachten Sie die Graphen dieser Funktionen und diskutieren Sie, welches Verhalten der Loesung Sie schon direkt aus der DGL erwarten konnten.
a) $x^2\ \ y\ \ '(x\ \ ) + (1 + x\ \ )\ y\ \ (x\ \ ) = 0,\ y\,(1) = 1$
b) $y\ \ (x\ \ )\ y\ \ '(x\ \ ) = e^x\ \ ,\ y\,(0) = 1$

Loesung

2 Differentialgleichungen erster Ordnung

2.1 Qualitatives Verhalten von Loesungen

Gewoehnliche Differentialgleichungen erster Ordnung koennen allgemein durch

y'(x) = f(x, y(x))

mit einer Funktion f : R^2 -> R angeben werden. Also ist durch die Differntialgleichung in jedem Punkt (in dem f definiert ist) die Steigung des Graphen der Loesung, wenn eine Loesung den Punkt *(x,y)* erreicht, gegeben. Der Befehl DEplot liefert das zugehoerige Richtungsfeld, d.h. in jedem Punkt *(x,y)* der Ebene einen Vektor in Richtung *(x,f(x,y))*. Betrachten wir z.B. folgende Differentialgleichung

```
> dgl:=D(y)(x)=2*(y(x))^2-y(x);
```

Wenn Sie nicht im ersten Kapitel schon die Toolbox aktiviert haben, muessen Sie dieses jetzt nachholen.

```
> with(DEtools):
```

(Der Doppelpunkt nach dem Befehl bewirkt, dass die Ausgabe des Ergebnisses, hier eine Liste der neuen Befehle, unterdrueckt wird). Nun koennen wir das Richtungsfeld zu dieser DGL betrachten

```
> DEplot(dgl,y(x),x=-2..4,y=-1..1);
```

Das Verhalten einer konkreten Loesung ist aus dem Bild zu erahnen. Offensichtlich existieren Loesungen, die gegen einen endlichen Grenzwert konvergieren. Da in diesem Fall *y'(t) ->0* fuer t -> ∞ konvergieren muss, ergeben sich moegliche Grenzwerte aus den Nullstellen der rechten Seite.

```
> solve(2*y^2-y=0,y);
```

Durch zusaetzliche Angabe einer Anfangsbedingung kann auch eine konkrete Loesung (oder mehrere) gezeigt werden.

```
> DEplot(dgl,y(x),x=-5..5,[0,0.2]);
```

Beachten Sie die Syntax dieses Befehls, die leider verschieden ist vom *dsolve* Befehl. Die Gruende dafuer liegen in weiteren Moeglichkeiten des *DEplot* Befehls

im Fall von DGL hoeherer Ordnung oder Systemen von DGL, auf die hier nicht weiter eingegangen werden kann.

Betrachten wir ein weiteres Beispiel.

```
> dgl:= D(y)(x)=sqrt(y(x));
> DEplot(dgl,y(x),x=-2..2,y=0..4);
```

Auch hier ist das Verhalten von moeglichen Loesungen deutlich zu sehen. Die Punkte *(x,0)* sind dabei aber kritisch, da ausgehend von einem solchen Anfangswert zwei Loesungen, die triviale Loesung *y(x)=0* und die entsprechend ansteigende Funktion moeglich sind.

```
> DEplot(dgl,y(x),x=-2..2,[-2,0],[-2,0.001]);
```

Beachten Sie die Moeglichkeit auch mehrere Anfangsbedingungen anzugegeben. Die ansteigende Loesung im Fall *y(-2)=0* wird von DEplot nicht gesehen, da der Graph numerisch bestimmt wird (s. Abschnitt 2). Mit

```
> dsolve(dgl,y(-2)=0,y(x));
```

erhalten wir hier nur eine Loesung. An solchen Verzweigungsstellen ist also Vorsicht geboten.

Bemerkung: In diesem Beispiel sind die Bedingungen des *Satzes von Picard-Lindeloef* in *y=0* verletzt, sodass keine eindeutige Loesbarkeit des Anfangswertproblems zu erwarten ist. Der *Satz von Peano* garantiert aber die Existenz einer Loesung.

Nun betrachten wir noch ein letztes Beispiel, die Phasengleichung zur Van-der-Pol Differentialgleichung.

```
> dgl:=D(v)(t)=(1-t^2)-t/v(t);
> dsolve(dgl);
```

Was ist passiert? Maple kann zu dieser Differentialgleichung keine Loesung bestimmen. Es ist uebrigens auch keine geschlossene Darstellung der Loesung bekannt. Betrachten wir nun das Richtungsfeld, wobei noch von der Moeglichkeit zusaetzlicher Optionen zum Befehl *DEplot* Gebrauch gemacht wird.

```
> DEplot(dgl,v(t),t=-2..2,[0,1],v=-2..2,linecolor=blue);
```

Das Richtungsfeld laesst sich natuerlich trotzdem angeben. Aber der numerischen Loesung des Anfangswertproblems duerfen wir nur im Intervall [-0.5,1.4] trauen, da die Steigung singulaer wird. (Sie koennen das Bild vergroessern, indem Sie es mit der linken Maustaste anklicken und eine der Markierungen mit der Maus nach aussen ziehen.)

Aufgabe: Diskutieren Sie anhand des Richtungsfeldes das Verhalten von Loesungen der DGL

$$y'(x) = x + \sin(y(x))$$

und erstellen Sie mit *DEplot* eine Ausgabe, die die Loesungen zu den Anfangsbedingungen

$$y(0)=-1,\ y(0)=0 \quad \text{und } y(0)=1$$

ohne das Richtungsfeld zeigt.

Loesung

2.2 Das Euler-Verfahren

Beim Befehl *DEplot* werden Loesungen zu Anfangswertproblemen numerisch approximiert. Es gibt eine Vielzahl von Verfahren, aber das einfachste ist sicherlich das Euler-Verfahren.
Nach dem Mittelwertsatz gilt fuer eine differenzierbare Funktion $y(x)$ die Identitaet

$$y(x+h) = y(x) + y'(z)\ h$$

mit einem z im Intervall $[x,x+h]$. Wenn nun y der Differentialgleichung

$$y'(x) = f\ (\ x,\ y(x)\ )$$

genuegt und $y(x)$ bekannt ist, so koennen wir versuchen fuer kleine Werte von h den Funktionswert $y(x+h)$ zu approximieren, indem die Ableitung an einer Zwischenstelle durch die Ableitung in x ersetzt wird

$$y(x+h) \quad y(x) + f(\ x,\ y(x)\ )\ h\ .$$

Starten wir von einem Anfangswert, z.B. $y(0)=1$, und legen wir eine *Schrittweite* $h>0$ fest, so koennen wir sukzessive Naeherungen an die Funktionswerte an den Stellen $0+nh$ fuer natuerliche Zahlen n bestimmen. Dies ist das Euler-Verfahren. Wir koennen es in Maple durch eine kleine Prozedur realisieren.

```
> myeuler:= proc(f,x0,y0,xN,h)
> local x,y,yN,seq:
> global eulerlist:
> y:=y0:
> seq:=NULL:
> for x from x0 by h to xN do
> yN  := y:
> seq := seq, [x,y]:
> y   := evalf(y+h*f(x,y)):
> od:
> eulerlist := [seq]:
> yN;
> end:
```

Die Eingabevariablen sind

f - die Funktion f in zwei Variablen
x0 - die Anfangsstelle

y0 - der Anfangswert
xN - der Endpunkt
h - die Schrittweite

Darueberhinaus werden noch lokale Variable, *x, y, yN, seq* und eine globale Variable *eulerlist*, die nach Ablauf des Programms eine Liste aller Approximationen enthaelt, definiert. Um den Zeilenumbruch bei der Eingabe ohne Auswertung des Kommandos zu erreichen, muessen Sie "shift" und "return" verwenden.

Nun kann das Verfahren ausgetestet werden. Betrachten wir etwa die DGL

$$y'(x) = e^x \quad y(x) \ .$$

```
> f:=(x,y) -> exp(x)*y;
```

Eine Approximation der Loesung an der Stelle *x=1* zum Anfangswert *y(0)=1* erhalten wir durch

```
> myeuler(f,0,1,1,0.1);
```

Vergleichen wir dieses Resultat mit der exakten Loesung:

```
> dgl:=D(y)(x)= f(x,y(x));
> lsg:=dsolve(dgl, y(0)=1,y(x));
> y:=unapply(rhs(lsg),x); evalf(y(1));
```

Naja, die Schrittweite muss wohl verkleinert werden. Zunaechst speichern wir aber noch die erste Approximation fuer ein spaeteres Bild.

```
> euler1:=plot(eulerlist,color=blue):
> myeuler(f,0,1,1,0.01);
```

Schon besser! Um uns die verschiedenen Approximationen in einem Bild anzusehen, laesst sich der Befehl display nutzen. (Dazu muss aber noch das Paket plots geladen werden.)

```
> euler2:=plot(eulerlist,color=red):
> exakt:=plot(rhs(lsg),x=0..1,color=black):
> with(plots): display([euler1,euler2,exakt]);
```

Selbstverstaendlich gibt es bessere numerische Verfahren als das Euler-Verfahren. Eine der gaengigsten Methoden, das *Runge-Kutta-Verfahren,* basiert auf einer geschickten Modifikation der hier aufgezeigten Idee und ist in Maple implementiert als Option zum *dsolve*-Befehl.

```
> y:='y': rk:=dsolve(dgl,y(0)=1,y(x),numeric);
```

Beachten Sie, dass zunaechst die zuvor als feste Funktion definierte Variable *y* durch die Eingabe *y:='y'* frei verfuegbar gemacht werden musste.
Maple erstellt eine Prozedur, die die approximierte Loesungsfunktion beinhaltet. Wir erhalten den Funktionswert an der Stelle *x=1* durch

```
> rk(1);
```

Einen plot dieser Loesung koennen wir uns mit dem Befehl odeplot aus dem DEtools Paket ansehen.

```
> odeplot(rk,[x,y(x)],-1..1);
```

Aufgabe: Schreiben Sie eine Prozedur, die das modifizierte Euler-Verfahren

$$y(x+h) \quad y + 1/2\,h \quad (\, f(x,y) + f(\, x+h,\, y+h\, f(x,y)\,)\,)$$

durchfuehrt (mit y auf der rechten Seite sei y(x) bezeichnet).
Vergleichen Sie dieses neue Verfahren mit dem oben definierten Euler-Verfahren am Beispiel der Differentialgleichung

$$y'(x) = e^x \quad y(x)$$

mit dem Anfangswert $y(0)=1$ auf dem Intervall $[0,1]$. Welches Vorgehen erscheint Ihnen als besser?

Loesung

2.3 Wachstums- und Zerfallsprozesse

Anwendungen von gewoehnlichen Differentialgleichungen begegnen wir heute in nahezu allen Wissenschaften. Eines der einfachsten dabei auftretenden Modelle dient zur Beschreibung von Wachstums- oder Zerfallsprozessen.

Zum Beispiel bestaetigen eine Vielzahl von biologischen Versuchen unter optimalen Bedingungen (Nahrung, Temperatur, etc.) exponentielles Wachstum von Bakterienkulturen. Wenn mit einer Population $p0$ zu einem Zeitpunkt $t0$ ein Versuch gestartet wird, so ist nach einer Zeiteinheit eine Vergroesserung der Population um eine Rate r zu beobachten, d.h. $p(t_0+1) = rp(t)$. Das einfachste Modell geht nun von konstanter Reproduktionsrate aus. Also ist die zeitliche Aenderung der Population, $p'(t)$, durch die lineare DGL

$$p'(t) = r\,p(t) \quad \text{mit Anfangsbedingung } p(t0) = p0$$

gegeben. Bestimmen wir zunaechst Schritt fuer Schritt die Loesung dieser separablen DGL:

```
> dgl:=D(p)(t)=r*p(t);
```

Unter der Annahme, dass p(t) nicht verschwindet kann durch p dividiert werden

```
> gln := dgl/p(t);
```

und Integration beider Seiten liefert

```
> ls:= int(lhs(gln),t); rs:= int(rhs(gln),t) + c;
```

wobei noch eine moegliche Integrationskonstante c beruecksichtigt werden muss. Offensichtlich fuehrt eine Substitution auf der linken Seite zum Ziel. Um dies hier explizit durchzufuehren, laesst sich der Befehl changevar aus der Toolbox student nutzen.

```
> with(student,changevar):  changevar(p(t)=x,ls);
> gln:=subs(x=p(t),%)=rs;
> lsg:=solve(gln,p)(t);
```

```
> c:=solve(subs(t=t0,lsg)=p0,c);
> lsg:=simplify(lsg);
```

Diese bekannten Schritte zur Loesung einer separablen DGL erhalten wir bequemer durch den *dsolve* Befehl:

```
> lsg:=rhs( dsolve(dgl,p(t0)=p0,p(t)) );
```

Setzen wir z.B. *t0=0* und *p0=100* so koennen wir uns in Abhaengigkeit von dem Parameter r verschiedene Entwicklungen ansehen:

```
> l1 := subs(t0=0,p0=100,r=1,lsg):
> l2 := subs(t0=0,p0=100,r=1.2,lsg):
> plot([l1,l2],t=0..2);
```

Eine wichtige Kennzahl ist die Verdoppelungsrate, d.h. in welchem Zeitraum sich die Population verdoppelt. Gesucht ist also die Loesung h der Gleichung

$$p(t+h) = 2\ p(t)$$

```
> solve(subs(t=x,lsg)=2*lsg,x) - t:  h:=simplify(%);
```

Wir sehen, dass diese Groesse wirklich von t unabhaengig ist und somit eine sinnvolle Kennzahl ergibt. Analog erhalten wir aus der Gleichung

$$p(t+h) = 1/2 \quad p(t)$$

bei Zerfallsprozessen die sogenannte *Halbwertzeit*.

Selbstverstaendlich sind die idealisierten Bedingungen in diesem Modell haeufig realitaetsfern. Um ein besseres Modell zu bekommen zum Beispiel um Futtermangel oder Krankheiten zu modellieren, muessen Wachstumsraten zugelassen sein, die vom Zeitpunkt und der aktuellen Groesse der Population abhaengen. Also ist $r = r(t,\ p(t))$. Ein erster Ansatz unter der Annahme, dass bei kleinen Populationen annaehernd optimale Bedingungen herrschen mit einer Rate R aber mit wachsender Population die Vermehrungsrate kleiner wird (evtl. sogar negativ), ist die Funktion

$$r(\ p(t)\ ) = R - b\ p(t).$$

Wir erhalten die separable nichtlineare DGL

$$p'(t) = R(1 - b\ p(t))\ p(t)$$

```
> dgl:= D(p)(t)=R*(1-b*p(t))*p(t);
> lsg:=rhs(dsolve(dgl,p(t0)=p0,p(t)));
```

Dieses kompliziertere Verhalten wird *logistisches Wachstum* genannt. Nun koennen wir das Verhalten fuer verschiedene Werte der Parameter untersuchen, zum Beispiel

```
> l1 := subs(t0=0,p0=1,R=1,b=0.1,lsg):
> l2 := subs(t0=0,p0=5,R=5,b=0.1,lsg):
> l3 := subs(t0=0,p0=12,R=1,b=0.1,lsg):
> plot([l1,l2,l3],t=0..10);
```

Wir beobachten zunaechst (fuer kleine t) ein exponentielles Wachstum (bzw. Abfall) und fuer grosse Zeiten konvergieren alle Loesungen (unabhaengig von

p0 und r aber abhaengig von b) gegen denselben Grenzwert (die stationaere Loesung).

```
> assume(R>0); limit(lsg,t=infinity);
```

Beachten Sie, dass ohne die zusaetzliche Bedingung an R durch den Befehl assume, hier kein Grenzwert ermittelt werden kann.

Aufgabe:
Der Zerfall des radioaktive Isotops Caesium-137 kann durch die DGL

$$y'(t) = -r\ y(t)$$

mit einer Zerfallsrate $r>0$ modelliert werden, wobei $y(t)$ die Anzahl der Atome zu einem Zeitpunkt t bezeichne. Durch Zerstrahlung verliert Caesium-137 ca 2.3
a) Bestimmen Sie die Zerfallsrate r und die Halbwertzeit von Caesium-137
b) Wenn jaehrlich durchschnittlich g Gramm Caesium-137 in die Atmosphaere ausgestossen wird, laesst sich dies durch die inhomogene DGL

$$y'(t) = -r\ y(t) + g$$

modellieren. Ermitteln Sie in Abhaengigkeit von g die Belastung der Atmosphaere, die sich nach einer laengeren Zeitspanne einstellen wird.

Loesung

3 Differentialgleichungen zweiter Ordnung

3.1 Der Stossdaempfer

Die Auslenkung einer Masse m an einer Feder aus ihrer Ruhelage, L, sei durch y(t) bezeichnet. Nach dem zweiten Newtonschen Gesetz gilt, dass das Produkt aus Masse und Beschleunigung die Kraft ist,

$$m\ y''(t) = F(\ t,\ y(t),\ y'(t)).$$

Dabei setzt sich die Kraft aus mehreren Anteilen zusammen, $F=F1+F2+F3+F4$ mit
Gravitation: $F1 = m\ g$ (g Erdbeschleunigung)
Rueckstellkraft: $F2 = -k\ (y(t)+L)$ ($k>0$ Federkonstante)
Daempfung: $F3 = -b\ y'(t)$ ($b>0$ Daempfungskonstante)
Externe Kraefte: $F4 = f(t)$.
Da sich die Gravitation, F1, und die Rueckstellkraft in Ruhelage, $F2 = -kL$, ausgleichen, erhalten wir die lineare inhomogene DGL zweiter Ordnung (mit konstanten Koeffizienten)

$$m\ y''(t) + b y'(t) + k y(t) = f(t)$$

Fuer Loesungsmethoden und die Loesungstheorie zu linearen DGL zweiter Ordnung sei auf die Literatur verwiesen. Im folgenden soll das Verhalten einer solchen Feder unter verschiedenen Bedingungen analysiert werden.
Zunaechst raeumen wir auf, laden die noetigen Toolboxen und definieren die DGL

```
> restart: with(plots): with(DEtools):
> dgl:= (D@@2)(y)(t) + b/m*D(y)(t) + k/m*y(t) = f(t)/m;
```

a) Die ungedaempfte (*b=0*), freie (*f=0*), Schwingung:

```
> f:=t->0; dgl1:=subs(b=0,dgl);
> dsolve(dgl1);
```

Wie zu erwarten ergibt sich eine harmonische Schwingung mit Frequenz $sqrt\left(\frac{k}{m}\right)$. Mit Festlegung einer Anfangsauslenkung und einer Anfangsgeschwindigkeit erhalten wir die eindeutig bestimmte Loesung, zum Beispiel

```
> abed:= y(0)=0, D(y)(0)=-1;
> dsolve(dgl1,abed,y(t));
```

b) Die gedaempfte, freie Schwingung:
Im folgenden legen wir *m=1* und *k=1* fest und betrachten eine Daempfungskonstante $b = 1/10$.

```
> dgl2:=subs(b=1/10, m=1, k=1, dgl);
> lsg:=rhs(dsolve(dgl2,abed,y(t)));
> plot(lsg,t=0..50);
```

Erhoehen wir die Daempfung der Feder so erreichen wir bei b=1 einen kritischen Bereich.

```
> dgl3:=subs(b=1, m=1, k=1, dgl);
> lsg:=rhs(dsolve(dgl3,abed,y(t))); plot(lsg,t=0..50);
```

Die Feder schwingt einmal durch und kommt wieder in die Ruhelage zurueck. Eine weitere Erhoehung der Daempfung fuehrt in den uebergedaempften Bereich wie bei einem intakten Stossdaempfer.

```
> dgl4:=subs(b=2, m=1, k=1, dgl);
> lsg:=rhs(dsolve(dgl4,abed,y(t))); plot(lsg,t=0..50);
```

c) Erzwungene Schwingungen:
Es sei angenommen, dass eine zusaetzliche Kraft auf die Feder wirkt, zum Beispiel

```
> f:=t->cos(a*t);
```

Mit einer Frequenz $a = 1/2$ erhalten wir folgende Loesung

```
> dgl4:=subs(a=1/2, b=1/10, m=1, k=1, dgl);
> lsg4:=dsolve(dgl4,abed,y(t)): lsg4:=collect(lsg4,exp);
```

Uff! - mit dem Befehl collect ist uebrigens der exponentielle Anteil ausgeklammert worden. Der stationaere Teil der Loesung (also ohne den exponentiellen Anteil) laesst sich mit dem Befehl remove aus diesem Ausdruck extrahieren.

```
> st_lsg4:= remove(has,rhs(lsg4),exp(-1/20*t)):
```

Nun lassen sich die Graphen der beiden Schwingungen vergleichen

```
> pe:=plot(rhs(lsg4), t=0..80, color=red):
> ps:=plot(st_lsg4, t=0..80, color=blue):
> display([pe,ps]);
```

Nach einer gewissen Zeit verhaelt sich also die erzwungene Schwingung wie die stationaere Loesung.
Das Verhalten von Loesungen in Hinblick auf die Frequenz a der anregenden aeusseren Kraft soll nun noch genauer untersucht werden.

```
> dgl5:=subs(m=1, k=1, dgl);
> lsg5:=dsolve(dgl5);
```

Es interessiert die Amplitude der stationaeren Loesung (der erste Summand).

```
> lsg_part:= subs(_C1=0,_C2=0,rhs(lsg5));
> tmax:= solve(diff(lsg_part,t)=0,t);
> amp:=subs(t=tmax,lsg_part);
```

und das Maximium der Amplitude in Abhaengigkeit von b ergibt sich aus

```
> amax:= solve(diff(amp,a)=0,a);
```

Aus diesen Berechnungen folgt die Resonanzfrequenz

```
> ares:=amax[2];
```

und die Loesungen im Resonanzfall:
a) mit Daempfung

```
> dgl6:=subs(a=ares,b=1/10,m=1, k=1, dgl);
> lsg:=rhs(dsolve(dgl6,abed,y(t))):  plot(lsg,t=0..80);
```

b) ohne Daempfung

```
> dgl7:=subs(a=ares,b=0,m=1, k=1, dgl);
> lsg:=rhs(dsolve(dgl7,abed,y(t))):  plot(lsg,t=0..80);
```

Unser Modell zeigt also das bekannte katastrophale Verhalten von schwingenden Systemen im Resonanzfall.

Nur in Spezialfaellen wie der oben beschriebenen DGL mit konstanten Koeffizienten lassen sich geschlossene Loesungen zu DGL hoeherer Ordnung angeben. Eine weitere Klasse sind Eulersche Differentialgleichungen, wie zum Beispiel

$$x^2\ y''(x) + x\ y(x) + a\ y(x) = 0\ .$$

Aufgabe:
a) Bestimmen Sie Loesungen zur Eulerschen DGL auf dem Intervall $(0,\infty\ )$ und diskutieren Sie das Verhalten von Loesungen fuer $x \mapsto 0$ und fuer $x \mapsto \infty$.
b) Klaeren Sie den Zusammenhang der gegeben Euler-DGL mit einer DGL mit konstanten Koeffizienten durch die Variablensubstitution $x = e^t$. (Hinweis: Verwenden Sie den Befehl Dchangevar)

Loesung

3.2 Der Paukenschlag zum Schluss

```
> restart: with(plots): with(DEtools):
```

Abschliessend soll ein kompletes Anfangsrandwertproblem behandelt werden. Fuer uns "Pauker" waehlen wir eine Pauke.

Die Membran der idealen Pauke sei der Einheitskreis. Die Auslenkung der Membran in Abhaengigkeit von der Zeit t und dem Ort (x,y) sei durch u(t,x,y) gegeben. Bei entsprechender Normierung aller physikalisch relevanten Parameter genuegt diese Funktion der Wellengleichung

$$(D_{1,1})(u)(t,x,y) - (D_{2,2})(u)(t,x,y) - (D_{3,3})(u)(t,x,y) = 0.0$$

Da die Membran fest eingespannt ist erhalten wir noch eine Randbedingung, zum Beispiel

$$u(t,x,y) = 0 \quad \text{fuer } x^2 + y^2 = 1 \quad .$$

Eine Anfangsbedingung zur Zeit t=0, zum Beispiel ein Impuls durch einen Schlag, der die Membran in Schwingungen versetzt, lassen wir zunaechst ausser Acht.

Die partielle DGL laesst sich in Maple wie folgt eingeben:

```
> pde:= D[1,1](u)(t,x,y)
> - D[2,2](u)(t,x,y) - D[3,3](u)(t,x,y) = 0;
```

Als ersten Schritt versuchen wir die Zeitabhaengigkeit zu separieren, d.h. es sind Loesungen in der Form

$$u(t,x,y) = w(t)\,v(x,y)$$

gesucht. Dazu laesst sich der Befehl *Dchangevar* nutzen.

```
> pde:=Dchangevar(u(t,x,y)=w(t)*v(x,y),pde,t);
```

Unter der Annahme, dass w und v nicht verschwinden, sortieren wir die Terme.

```
> pde:=collect(pde/w(t)/v(x,y),D);
```

Offensichtlich ist der erste Summand vom Ort und der Rest von der Zeit unabhaengig. Also muessen diese Ausdruecke konstant sein. Wir erhalten zwei DGL'n, die nur durch diese Konstante gekoppelt sind.

```
> dgl_t:=remove(has,op(1,pde),x)=-k^2;
> dgl_xy:=remove(has,op(1,pde),t)=k^2;
```

Hier wurde schon vorgegriffen mit der Annahme k>0, die sich aber aus der ersten DGL ergibt, wenn wir nur beschraenkte Loesungen in der Zeit zulassen, die zum Zeitpunkt t=0 verschwinden (die Pauke soll aus der Ruhelage angeregt werden).

Betrachten wir zunaechst die Zeitabhaengigkeit:

```
> lsg_t:=dsolve(dgl_t,w(0)=0,w(t));
```

Da keine Daempfung im Modell beruecksichtigt ist, erhalten wir ein harmonisches Schwingungsverhalten in der Zeit mit einer Frequenz k.

Nun die Ortsabhaengigkeit - es handelt sich uebrigens um die Helmholtzgleichung.

```
> dgl_xy:= dgl_xy*v(x,y);
```

Wegen der Geometrie ist es sinnvoll, Polarkoordinaten fuer diese DGL zu verwenden.

```
> dgl_p:=PDEchangecoords(dgl_xy,[x,y],polar,[r,phi]);
```

Dieser Befehl soll in zukuenftigen Versionen von Maple ueberarbeitet werden. Es scheint, dass eine verbesserte Variante der Variablen Transformation *Dchangevar* geplant ist. Im letzten Term sollte eigentlich v in Abhaengigkeit von r, ϕ erscheinen. Wir korrigieren dies durch

```
> dgl_p:=subs(v(r*cos(phi),r*sin(phi))=v(r,phi),dgl_p);
```

Im naechsten Schritt wird wieder ein Separationsansatz,

$$v(r, \phi\ ) = rad(r)\ arg(\phi\ ),$$

versucht.

```
> dgl_p:=Dchangevar(v(r,phi)=rad(r)*arg(phi),dgl_p,r);
> dgl_p:=collect(r^2*dgl_p/rad(r)/arg(phi),D);
```

Die beiden unabhaengigen Terme lassen sich wieder getrennt voneinander untersuchen.

```
> dgl_phi:=remove(has,op(1,dgl_p),r)=c;
```

und loesen diese DGL

```
> lsg_phi:=dsolve(dgl_phi,arg(phi));
> arg:=unapply(rhs(lsg_phi),phi);
```

Wir benoetigen nun alle $2\,\pi$ -periodischen Loesungen. Um alle Loesungen, die *solve* findet, zu bekommen, muss die Umgebungsvariable *_EnvAllSolutions* umgesetzt werden.

```
> _EnvAllSolutions:=true:
> lsg_c:=solve(arg(0)=arg(2*Pi),D(arg)(0)=D(arg)(2*Pi),c);
```

Die Schlange hinter $\_Z$ bedeutet, dass eine Bedingung an $\_Z$ gestellt ist. Diese laesst sich mit dem Befehl about ermitteln, wobei durch indets die unbestimmte Variable $\_Z$ aus dem Ausdruck extrahiert wird.

```
> about(indets(subs(lsg_c[2],c)));
```

Also setzen wir $c = n^2$ mit einer natuerlichen Zahl n.

```
> assume(n,integer); assume(n>=0); c:=n^2;
```

und erhalten die Loesung

```
> lsg_phi:=simplify(arg(phi));
```

Nun bleibt nur noch die radiale Abhaengigkeit zu klaeren. Wir erhalten die Differentialgleichung:

```
> dgl_r:=simplify(eval(dgl_p))*rad(r);
> dgl_r:=collect(dgl_r,rad(r));
```

Dies ist eine Besselsche Differentialgleichung. Zum Glueck kennt Maple die allgemeine Loesung, die sonst durch einen verallgemeinerten Potenzreihenansatz bestimmt werden muesste.

```
> lsg_rad:=dsolve(dgl_r,rad(r));
```

Schauen wir uns doch zunaechst einige Besselfunktionen an

```
> J0:= plot(BesselJ(0,t), t=0..10, color=red):
> J1:= plot(BesselJ(1,t), t=0..10, color=blue):
> J2:= plot(BesselJ(2,t), t=0..10, color=green):
> display(J0,J1,J2);
```

und nun noch die Neumannfunktionen

```
> Y0:= plot(BesselY(0,t), t=0..10, y=-1..1, color=red):
> Y1:= plot(BesselY(1,t), t=0..10, color=blue):
> Y2:= plot(BesselY(2,t), t=0..10, color=green):
> display(Y0,Y1,Y2);
```

Diese Loesungen sind singulaer und kommen daher fuer unsere Pauke nicht in Betracht. Also muss die Konstante *_C2 =0* sein und der radiale Anteil ist durch die Besselfunktionen gegeben. Weiter interessieren hier nur reelle Loesungen, sodass die Annahme k>0 noch mit berueckichtigt werden kann.

```
> assume(k>0):
> lsg_rad:= rhs(subs(_C2=0,lsg_rad));
```

Bisher ist die Randbedingung $u = 0$ auf dem Rand der Pauke nicht in die Loesung integriert. Dies ist nur zu erreichen, wenn $rad(1) = 0$ gilt.

```
> solve(subs(r=1,lsg_rad)=0,k);
```

Offensichtlich sind alle unendlich vielen Nullstellen der n-ten Besselfunktion Loesungen dieser Gleichung (s. plot). Der Befehl BesselJZero liefert uns Approximationen an diese Nullstellen. Zum Beispiel,

```
> z01:=evalf(BesselJZeros(0,1));
> z11:=evalf(BesselJZeros(1,1));
> z21:=evalf(BesselJZeros(2,1));
> z02:=evalf(BesselJZeros(0,2));
> z12:=evalf(BesselJZeros(1,2));
```

$$z01 := 2.404825558$$
$$z11 := 3.831705970$$
$$z21 := 5.135622302$$
$$z02 := 5.520078110$$
$$z12 := 7.015586670$$

Insgesamt haben wir nun, dass Loesungen des Anfangsrandwertproblems durch entsprechend konvergente Reihen der Form

$$u(t, r, \phi) = \sum_{n=1}^{\infty} (an \sin(m\phi) + bn \cos(m\phi)) J(m, zn\, r) \sin(zn\, t)$$

gegeben sind, wobei *zn* die der Groesse nach nummerierten Nullstellen der Besselfunktionen sind und *m(n)* den entsprechenden Index der zugehoerigen Besselfunktion bezeichnet. Es laesst sich zeigen, dass so alle Loesungen beschrieben werden koennen, wobei sich die Konstanten *an, bn* aus der Anfangsbedingung

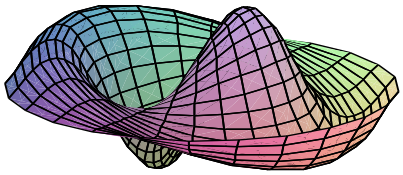
ergeben. Es ist zu sehen, dass der Ton einer Pauke sehr viel komplizierter ist, als der einer schwingenden Saite (s. Aufgabe), da die auftretenden Frequenzen, $2\,\frac{\pi}{zn}$, in keinem rationalen Verhaeltnis stehen.
Sehen wir uns nun noch Terme (Modes) der Reihe, also die Grund- und Oberschwingungen der Pauke an, zum Beispiel der Mode (1,2):

```
> u3:= cos(phi)*BesselJ(1,z12*r)*sin(z12*t);
```

$$u3 := \cos(\phi)\, BesselJ(1, 7.015586670\, r) \sin(7.015586670\, t)$$

Mit der dreidimensionalen Version des Befehls *animate*, animate3d, laesst sich nun dieser Mode unserer Pauke bewundern. Klicken Sie dazu nach der Ausfuehrung des Befehls das Bild mit der linken Maustaste an. In der dann erscheinenden zusaetzlichen Befehlsleiste oben starten Sie die Animation durch den Pfeil-Button. Testen Sie auch die weiteren Moeglichkeiten, die dort zur Verfuegung gestellt werden.

```
> animate3d([r*cos(phi), r*sin(phi), 0.3*u3], r=0..1, phi=0..2*Pi,
t=1..1.8, frames=20);
```



Interessant sind auch die Knotenlinien eines Modes, also die Punkte auf der Membran, die sich bei einer solchen Schwingung nicht bewegen. Diese lassen sich durch die Aufsicht auf einen Konturplot zu einem festen Zeitpunkt, der nur die 0 Niveaulinien enthaelt, bekommen.

```
> plot3d([r*cos(phi), r*sin(phi), 0.5*subs(t=1,u3)], r=0..1, phi=0..2*Pi,
style=contour, contours = [0], orientation=[0,0]);
```

Wie waere es nun mit eigenen Versuchen zur schwingenden Saite. Aber Vorsicht - es kann schnell viel Zeit beim Experimentieren vergehen.

Aufgabe: (Die Schwingung einer fest eingespannten Saite)
Eine Saite habe die Laenge $2\,\pi$. Mit *u(x,t)* sei die Auslenkung zur Zeit *t>0* und im Ort *0¡x¡2* π bezeichnet. Diese Auslenkung genuegt der Wellengleichung

$$(D_{2,2})\,(u)\,(x,t) - a^2\,(D_{1,1})\,(u)\,(x,t) = 0$$

(wobei physikalische Parameter (Masse, Elastizitaet, etc.) wieder entsprechend normiert sind). Da die Saite fest eingespannt ist, erhalten wir die Randbedingungen u(0,t)=u(2 π ,t)=0.

Bestimmen Sie die Grundfrequenz und Obertoene der Saite in Abhaengigkeit von *a* und erstelllen Sie eine animierte Darstellung zum Beispiel eine freigewaehlte Ueberlagerung vom Grundton und ein oder zwei Obertoenen.

Loesung

4 Kurze Uebersicht

4.1 Liste der verwendeten speziellen Befehle zur Behandlung von gewoehnlichen Differentialgleichungen

Fuer die aufgelisteten Befehle mit Ausnahme von *dsolve* ist die Toolbox *DEtools* erforderlich .

Dchangevar Variablensubstitution in Differentialgleichungen
DEplot a) bei einer Dgl erster Ordnung wird das Richtungsfeld und/oder die numerische Loesung eines Anfangswertproblems gezeigt
b) Im Allgemeinen werden numerisch bestimmte Loesungskurven gezeigt (s. auch DEplot3d)
DEtools Programmpaket zu Differentialgleichungen
dsolve Befehl zum Loesen von gewoehnlichen Differentialgleichungen oder Systemen von Dgl. erster Ordnung. Sowohl allgemeine Loesungen als auch Loesungen zu Anfangswertproblemen werden analytisch oder numerisch berechnet.
odeadvisor Ermittlung der Klasse in die eine Differentialgleichung eingeordnet wird.
odeplot Plot-Befehl zu *dsolve* aus dem DEtools Paket
odetest Befehl zum Verifizieren, ob eine Funktion eine Differentialgleichung erfuellt.

4.2 Literatur

Aus der Vielzahl an Literatur zu gewoehnlichen Differentialgleichungen, ihrer numerischen Behandlung und zu Maple sei hier nur auf einige wenige hingewiesen.

Differentialgleichungen:
Ein allgemeines aber knapp gehaltenes Nachschlagewerk zu Differentialgleichungen ist

- D. Zwillinger, *Handbook of differential equations,* Academic Press, 1998.

Ein umfangreiches Lehrbuch, das viele Beispiele im Text und in Aufgaben beinhaltet, ist

- H. Heuser, *Gewoehnliche Differentialgleichungen,* 2.Aufl., Teubner Verlag, Stuttgart, 1991.

Ein Klassiker aus den USA ist

- W.E. Boyce and R.C. DiPrima, *Gewoehnliche Differentialgleichungen* Spektrum Verlag, Heidelberg, 1995.

Eine Einfuehrung in die numerische Behandlung von Differentialgleichungen findet sich in

- H.R. Schwarz, *Numerische Mathematik,* Teubner Verlag, Stuttgart, 4.Aufl. 1997.

Als Fortsetzung des letzten Abschnitts sei noch empfohlen

- N.H. Fletcher and T.D. Rossing, *The Physics of Musical Instruments,* Springer Verlag, New York, Heidelberg, 1991.

Maple:

Neben der allgemeinen Literatur zu Maple ist nuetzlich

- D. Nowottny, *Mathematik am Computer*, Springer Verlag 1999.

Ein umfangreicher Mathematikkurs (Niveau Grundstudium fuer Natur- und Ingenieurwissenschaftler) findet sich in den beiden Baenden

- W. Werner, *Mathematik lernen mit Maple,* dpunkt.verlag, 2.Aufl., 1998.

Das spezielle Thema, Behandlung von Differentialgleichungen mit Maple, wird dargestellt in

- M.L. Abell and J.P. Braselton, *Differential Equations with Maple,* Academic Press, New York, 1999.

An Ihren Anregungen, Korrekturen, Kritiken etc. zu diesem Arbeitsblatt bin ich sehr interessiert (E-mail: *hettlich@math.uni-karlsruhe.de*).
Die *mws*-Datei zum Worksheet liegt auf meiner homepage unter
http://www.mathematik.uni-karlsruhe.de/mi2kirsch/ hettlich/seite/maple_work