1 Differentialgleichungen mit Matlab lösen

Eine Differentialgleichung (DGL) ist eine Gleichung für eine gesuchte Funktion mit einer oder mehreren Variablen, in der auch Ableitungen dieser Funktion vorkommen. Man unterscheidet zwei verschiedene Arten von Differentialgleichungen, gewöhnliche und partielle DGL. Enthält die gesuchte Funktion nur eine Variable, so nennt man die Gleichung auch gewöhnliche Differentialgleichung.

$$\frac{\partial y(x)}{\partial x} = 2xy$$

Eine partielle DGL enthält partielle Ableitungen von mehreren Variablen. Z.B.

$$\frac{\partial y(x,t)}{\partial t} + \frac{\partial y(x,t)}{\partial x} = 0$$

Wir werden uns hier auf gewöhnliche DGL beschränken.

1.1 Analytische Lösung

Falls eine lineare Differentialgleichung eine analytische Lösung hat, kann man sie in Matlab mit der Funktion dsolve explizit lösen.

DGL der 1. Ordnung

Will man $y' = 2xy$ lösen, so schreibt man:

```
y=dsolve('Dy=2*x*y','x')
y =
C2*exp(x^2)
```

- Die Gleichung muss als String übergegeben werden. Dabei wird das Differential als Dy bezeichnet.
  Die unabhängige Variable x, muss man ebenfalls als String angeben.
- Die Ausgabe ist vom Typ *symbolic*. Schauen Sie sich die Matlab-Hilfe an, wenn Sie mehr zu diesem Datentyp wissen wollen.
- $C*$ ist immer eine Integrationskonstante.

DGL der 1. Ordnung mit Anfangsbedingung

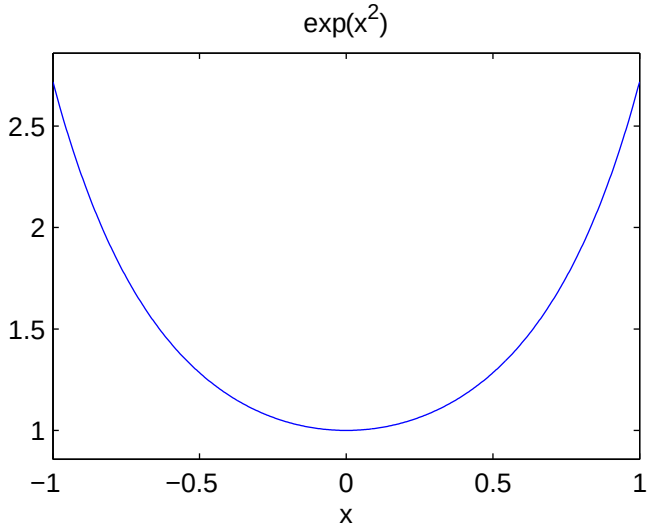
Hat man eine Anfangsbedingung, z.B. $y(0) = 1$ vorgegeben, so erhält man die explizite Lösung ohne Integrationskonstanten:

```
y=dsolve('Dy=2*x*y','y(0)=1','x')
y =
```

exp(x^2)

Diese Lösung kann man auch graphisch mit der Funktion ezplot darstellen. Dabei muss man Intervallgrenzen für x angeben.

ezplot(y, [-1,1])



Möchte man den numerischen Wert von z.B A$y(0.5)$ wissen, tippt man

double(subs(y,'x',0.5))

DGL höherer Ordnung

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

y=dsolve('D2y + Dy =2*y', 'y(0)=1', 'Dy(0)=1', 'x')

$$y''' = x, \quad y(0) = 1,\ y(1) = 1,\ y(-1) = 1$$

y=dsolve('D3y =x','y(0)=0','y(1)=1','y(-1)=1','x')

System von DGL

$$y_1' = y_2; \quad y_2' = y_1, \quad \text{mit} \quad y_1(0) = 100;\ y_2(0) = 50$$

y=dsolve('Dy1 =y2','Dy2=-y1','y1(0)=100', 'y2(0)=50','x')

Die Ausgabe ist von Typ *structure*. Die beiden Ergebnisse erhält man folgendermaßen:

y.y1
y.y2

Man kann die beiden Funktionen auch zusammen darstellen. Leider kann man bei ezplot nicht auf der üblichen Weise die Farbe ändern.

```
ezplot(y.y1, [0 20])
hold on
h=ezplot(y.y2, [0,20]);
set(h, 'color', [1 0 0])
hold off
```

In Matlab kann man auch Richtungsfelder zeichnen, wenn Sie daran interessiert sind, suchen Sie einfach im Internet nach "Richtungsfeld Matlab" oder "slope field matlab"

1.2 Aufgaben

1. Lösen Sie die Differentialgleichung

$$y' = \frac{1}{1-x}y + x - 1 \quad \text{mit} \quad y(2) = 0$$

einmal mit und ohne Anfangswertbedingung. Sie können auch eine andere Anfangswertbedingung einsetzen.
Plotten Sie die Funktion mit a) ezplot und b) plot

1.3 Numerische Lösung

Matlab stellt zur numerischen Lösung bei Systemen gewöhnlicher Differentialgleichungen verschiedene Funktionen mit verschiedenen Eigenschaften zur Verfügung:

- ode45, Dormand-Rince Verfahren (Runge-Kutta)
- ode23, Bogacki-Shampine Verfahren (Runge-Kutta)
- ode113, Adams-verfahren
- ode15s, BDF Verfahren
- ode23s, Rosenbrock Verfahren
- ode23t, Trapez-Regel
- ode23tb, TR-BDF-Verfahren

Wir werden hier nur mit der Funktion ode45 arbeiten.
Die Syntax ist im Vergleich mit der analytischen Lösung sehr verschieden. Für jede Differentialgleichung muss man sich eine m.Funktion in Matlab schreiben.

[x,y] = ode45(@Fktname, Intervall, Anfangswerte, Optionen, FktInput);

1. **Funktionsname** Der Name einer m.Funktion, die im einfachsten Fall nur eine Gleichung beschreibt. Der Funktionskopf hat im einfachsten Fall die Form:

   function dy = Funktionsname(x,y)

   Die Argumente x und y müssen auch dann angegeben werden, wenn sie zur Berechnung der Ableitungen nicht benötigt werden. Diese beiden Argumente sind Spaltenvektoren. Der Funktionswert, hier dy genannt, muss die Ableitungen der Differentialgleichungen liefern. **dy muss ein Spaltenvektor sein**.

2. **x-Intervall**, ein Zeilen- oder Spalten-Vektor der Länge 2, der das Integrationsintervall festlegt.

3. **Anfangswerte** als Zeilen- oder Spalten-Vektor, der die Anfangswerte des Systems festlegt.

4. **Optionen** optional, Struktur, die die Eigenschaften von ode45 ändern. Siehe Hilfe ode45 und odeset. Wir werden die Eigenschaften nicht ändern.

5. **FktInput**, optional. Falls die m-Funktion noch zusätzliche Inputargumente benötigt, kann man diese hier angeben, siehe Beispiel 3.

Beispiel 1

Wir lösen jetzt noch mal erneut die DGL:

$$y' = 2xy, \quad \text{mit} \quad y(-1) = \exp(1)$$

Die Funktion muss man unter DGL1.m abspeichern.

```
function dy=DGL1(x,y)
dy=2*x*y;
```

Die Gleichung soll im Intervall [-1 1] gelöst werden. Den Anfangswert, exp(1), ist der Wert, der am Anfang gelten soll. Also $y(-1) = exp(1)$.

Das bedeutet, man kann nicht die Bedingung y(0)=1 angeben.

```
[x1,y1]=ode45(@DGL1,[-1 1],exp(1));
figure
plot(x1,y1)
```

Beispiel 2 - Räuber-Beute Ein klassisches Modell zur Beschreibung von zwei einander bedingenden Populationsgrößen sind die Lotka-Volterra Gleichungen. Es handelt sich um ein System aus zwei gekoppelten Differentialgleichungen:

$$\frac{dP}{dt} = P(t)(r_P + c_P R(t))$$
$$\frac{dR}{dt} = R(t)(r_R + c_R P(t))$$

mit

$P(t)$	Populationsgröße der Beute zum Zeitpunkt t
r_P	konstante Reproduktionsrate der Beute, wenn keine Räuber anwesend sind (> 0, da die ungestörte Population wächst)
c_P	Änderung der Beute-Population pro Räuber (< 0, da die Population durch jeden Räuber verkleinert wird)
$R(t)$	Populationsgröße der Räuber
r_R	konstante Reproduktionsrate der Räuber, wenn keine Beute anwesend ist (< 0, da die Räuber ohne Beute sterben)
c_R	Änderung der Räuber-Population pro Beutetier (> 0, da durch jedes Beutetier Geburten ermöglicht werden)

Wir simulieren jetzt das Populationswachstum mit diesen Differentialgleichungen. Dabei setzen wir $r_P = 0.08; c_P = -0.002; r_R = -0.2; c_R = 0.0004$; und schreiben die m-Funktion:

```
function dbr=raeuberbeute(t,br)
dbr=zeros(2,1);
rP=0.08;
cP=-0.002;
rR=-0.2;
cR=0.0004;
dbr(1)=rP*br(1)+cP*br(1)*br(2);
dbr(2)=rR*br(2)+cR*br(1)*br(2);
```

Wir schauen ins einen Zeitraum von 100 Tagen an. Die Anfangspopulation der Beute ist 500 und die der Räuber 20.

```
[x,rb]=ode45(@raeuberbeute,[0,100],[500 20]);
plot(x,rb(:,1))
hold on
plot(x,rb(:,2),'r')
legend('Beute','Raeuber')
xlabel('Zeit in Tage')
ylabel('Populationsgroesse')
hold off
```

Man kann auch die Populationsgrößen gegeneinander auftragen.

```
figure plot(rb(:,1),rb(:,2))
xlabel('Populationsgroesse der Beute')
ylabel('Populationsgroesse der Raeuber')
```

Beispiel 3 erweiterte Eingabeargumente

Es ist nartürlich lästig immer wieder eine ganze Datei zu ändern, falls man Werte in seiner m-Funktion variieren möchte. Wir schreiben eine erweiterte Funktion für das Räuber-Beute Beispiel.

```
function dbr=raeuberbeute2(t,br,werte)
dbr=zeros(2,1);
rP=werte(1);
cP=werte(2);
rR=werte(3);
cR=werte(4);
dbr(1)=rP*br(1)+cP*br(1)*br(2);
dbr(2)=rR*br(2)+cR*br(1)*br(2);
```

Damit können wir jetzt ganz einfach die Werte variieren.

```
rP=0.08;
cP=-0.002;
rR=-0.2;
cR=0.0004;
[x,rb]=ode45(@raeuberbeute2,[0,100],[500 20],[],[rP cP rR cR]);
```

Variieren Sie die Eingabeparameter!

1.4 Aufgaben

1. Lösen Sie $y' = \frac{2}{x}y + 2x^3$ mit $y(2) = 20$ analytisch und numerisch im Interval [2 20]. Unterscheiden sich die analytische und numerische Lösung?

2. **Hodgkin-Huxley-Model** Das Hodgkin-Huxley-Modell ist das berühmteste Modell zur Simulation von Neuronen.[1] Das Modell simuliert das Membranpotential und Ionenströme von Neuronen nahe an den biologischen Gegebenheiten.
   Der Gesamtmembranstrom ist die Summe der Kalium-, Natrium- und Leck- Ströme I_K, I_{Na} und I_L und des externen Stroms I_{ext} (z. B. ein durch eine externe Elektrode applizierter Strom), der die Membran auflädt. Das Membranpotential V gehorcht dann der Differentialgleichung

$$C\frac{dV}{dt} = I_{ext} - I_K - I_{Na} - I_L$$

   Die Ströme I_K, I_{Na} und I_L ergeben sich aus der Differenz der Membranspannung zum jeweiligen Gleichgewichtspotential E und der dazugehörigen Leitfähigkeit g:

$$\begin{aligned} I_K &= g_K \cdot (V - E_K) \\ I_{Na} &= g_{Na} \cdot (V - E_{Na}) \\ I_L &= g_L \cdot (V - E_L). \end{aligned}$$

   Die Leitfähigkeiten g_K und g_{Na} sind zeitabhängig und somit für die Entstehung eines Aktionspotentiales verantwortlich, während die Leckleitfähigkeit g_L konstant bleibt, da sich der Widerstand der Membran als unveränderlich annehmen lässt. Hodgkins und Huxleys führten Gatingvariablen (engl. Gate: Tor) ein, die die (probabilistischen) Dynamiken der Ionenkanäle nachbilden. Diese Variablen geben den Anteil der gerade geöffneten Kanäle an. Zusammen mit der Maximalleitfähigkeit $\bar{g}$ eines Ions beschreibt die Gatingvariable den gerade durch die Membran fließenden Strom:

$$\begin{aligned} I_K &= \bar{g}_K \cdot n^4 \cdot (V - E_K) \\ I_{Na} &= \bar{g}_{Na} \cdot m^3 \cdot h \cdot (V - E_{Na}) \end{aligned}$$

   Die drei Gatingvariablen sind folgenden Dynamiken unterworfen:

$$\begin{aligned} \frac{dn}{dt} &= \alpha_n(V) \cdot (1 - n) - \beta_n(V) \cdot n, \\ \frac{dm}{dt} &= \alpha_m(V) \cdot (1 - m) - \beta_m(V) \cdot m, \\ \frac{dh}{dt} &= \alpha_h(V) \cdot (1 - h) - \beta_h(V) \cdot h. \end{aligned}$$

[1] http://de.wikipedia.org/wiki/Hodgkin-Huxley-Modell

Zur weiteren Erklärung siehe auch z.B.
http://icwww.epfl.ch/ gerstner/SPNM/node14.html

(a) Auf der Internetseite
http://classes.engineering.wustl.edu/cse100b/fall10/index.php/Hodgkin_Huxley_Modeling_of_Action_Potentials
ist der Code für dieses Modell. Speichern Sie das Modell ab und führen Sie den Code aus.

(b) Ab welchen injizierten Strom werden Aktionspotentiale ausgelöst? Schreiben Sie dazu ein Programm, das systematisch die Stromstärke I_{ext} variiert und automatisch die Anzahl der Aktionpotentiale ermittelt.

(c) Variieren Sie die Natriumleitfähigkeit. Welchen Einfluss hat dieser Parameter auf die Schwelle des externen Stroms, ab dem Aktionspotentiale generiert werden?