MATHEMATISCHES INSTITUT
DER UNIVERSITÄT ZU KÖLN
Johannes Schropp, M. Kurth, A. Gail

WS 2003/2004
2.12.2003

Numerische Lösung von retardierten Differentialgleichungen mit Matlab

Das vorliegende Skript soll dazu dienen, selbstständig retardierte Differentialgleichungen mit Matlab zu lösen. Nach einer kurzen Einführung in Matlab wird im zweiten Abschnitt erläutert, wie man gewöhnliche Differentialgleichungen untersucht. Es ist sinnvoll, alle gegebenen Beispiele am Rechner nachzuvollziehen. Insbesondere die Aufgaben im letzten Abschnitt sollen dazu dienen, einen Leitfaden zum eigenständigen Arbeiten zu geben.

Ziel der praktischen Übung ist es **nicht**, alle Aufgaben zu lösen sondern - je nach Vorkenntnis - einen Einblick in Matlab zu erhalten. Mit Hilfe dieses Skriptes sowie der Dokumentation von Matlab selbst sollte es dann möglich sein, die Übungsaufgaben zu bearbeiten.

1 Grundlagen

1.1 Vorbereitungen

Vor dem erstmaligen Aufruf von Matlab sollten Sie sich ein Verzeichnis mit dem Namen `matlab` anlegen. Dazu geben Sie nach dem Anmelden am Rechner das Kommando

```
mkdir matlab
```

ein.

Vor jedem Aufruf von Matlab wechseln Sie mit

```
cd matlab
```

in dieses Verzeichnis.

1.2 Matlab starten

Durch Eingabe des Kommandos

```
MATLABPATH=/a/teaching/matlab/kuepper/fotisdde23 matlab
```

wird Matlab gestartet und direkt die Routine `dde23` eingebunden, die wir zum Lösen von retardierten Differentialgleichungen benötigen. Nach einigen Sekunden erscheint auf dem Bildschirm das Matlab-Fenster, welches wieder in drei Unterfenster aufgeteilt ist. Das momentan aktive Unterfenster ist mit einem blauen Balken am oberen Rand gekennzeichnet. Die Größe der Unterfenster lassen sich den individuellen Wünschen anpassen. Dazu klicken Sie mit der linken Maustaste auf den Rand zwischen den Fenstern und verschieben diesen mit der Maus (bei gedrückter linker Taste).

Für uns interessant ist zunächst nur das Unterfenster mit der Überschrift

Command Window .

Hier geben wir die Matlab-Befehle ein.

1.3 Matlab beenden

Um Matlab zu beenden, klicken Sie mit der linken Maustaste im MATLAB-Fenster den Menüpunkt **File** an und wählen dann (mit der linken Maustaste) in dem erscheinenden Untermenü den Punkt **Exit MATLAB** aus.

Alternativ können Sie im Command Window den Befehl `exit`

Beim Verlassen wird die aktuelle Matlab-Konfiguration (Größe und Anzahl der Unterfenster) gespeichert und erscheint so beim nächsten MATLAB-Aufruf wieder. Allerdings müssen Sie die Pfadkonfiguration noch in Ihr Verzeichnis `matlab` speichern. Dies geschieht, indem Sie unter dem Menüpunkt `file` und `SetPath` wählen und dann `save` anklicken und das Verzeichnis auswählen. Nun können Sie ab dem zweiten Aufruf MATLAB einfach mit `matlab` aufrufen, da der Pfad zu der Routine `de23` dann gesetzt ist.

1.4 Online-Hilfe

Eine komplette Dokumentation von MATLAB (in englischer Sprache) befindet sich auf dem Rechner. Diese wird gestartet durch Eingabe von

`helpdesk`

im Command Window. Es wird dann ein neues Fenster **Help** eröffnet. Um dieses Fenster wieder zu schließen, klicken wir mit der linken Maustaste den Menüpunkt **File** und dann den Unterpunkt **Close Help** an.

Ist man nur an Informationen zu einem bestimmten Matlab-Befehl interessiert, so gibt es zwei Möglichkeiten:

(i) Die Eingabe von

`doc` <*kommando*>

im Command Window liefert ausführliche Information in einem neuen Help-Window. Dabei bedeutet <*kommando*> ein Matlab-Kommando. Z.B. erhalten Sie durch

`doc inv`

Information über das Matlab-Kommando `inv`, welches die inverse Matrix berechnet.

(ii) Das Kommando

`help` <*kommando*>

gibt die Information im Command Window aus. Z. B. erhalten wir mittels

`help inv`

die folgende Ausgabe:

```
>> help inv

 INV    Matrix inverse.
    INV(X) is the inverse of the square matrix X.
    A warning message is printed if X is badly scaled or
    nearly singular.

    See also SLASH, PINV, COND, CONDEST, LSQNONNEG, LSCOV.

>>
```

1.5 Eingabe von Matlab-Kommandos

Die Eingabe von Matlab-Kommandos erfolgt im Command Window. Matlab arbeitet interaktiv, d.h. jeder Matlab-Befehl wird nach Betätigen der Eingabetaste sofort ausgeführt (im Gegensatz zu den Programmiersprachen wie Fortran, Pascal oder C, bei denen das komplette Programm zuerst übersetzt werden muß).

Dabei ist zu beachten:

- Es wird zwischen Groß- und Kleinbuchstaben unterschieden.
- Das Zeichen % dient als Kommentarzeichen: der Text rechts davon bis zum Zeilenende wird als Kommentar betrachtet (und somit nicht als Matlab-Befehl interpretiert).

- Ein Matlab-Befehl kann sich über mehrere Zeilen erstrecken, muß dann jedoch am Zeilende mit dem Zeichen ... markiert werden.
- In einer Zeile dürfen mehrere Matlab-Befehle stehen. Diese müssen jedoch durch einen Strichpunkt (bzw. ein Komma) getrennt werden.
- Ein Matlab-Kommando kann mit einem Strichpunkt abgeschlossen werden. Dann wird das Ergebnis dieses Befehls nicht im Command Window angezeigt. Ohne Strichpunkt am Ende erscheint das Ergebnis im Command Window.

Hier sind einige Beispiele für Matlab-Kommandos:

```
x = 1.5
X = 1e-7;
a = 1.5; b = 4.5; c = 3.8; s = (a+b+c)/3; % Mittelwert
u = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...
      -1/8 + 1/9 - 1/10 + 1/11 - 1/12;
u
```

`X` und `x` sind verschiedene Variablen. Die letzte Zeile bedeutet, daß der Wert der Variablen `u` im Command Window ausgegeben wird.

In Matlab können auch Linux-Kommandos eingegeben werden. Diese werden mit einem vorangestellten ! gekennzeichnet. Zum Beispiel wird durch

```
!ls -al
```

der Inhalt des aktuellen Verzeichnisses im Command Window aufgelistet.

2 Lösen von ODE's mit Matlab

Matlab bietet zwei Funktionen an, welche Anfangswertaufgaben mit Hilfe von expliziten Runge-Kutta-Verfahren lösen. Im Standardfall verwendet man `ode45`. Ist nur eine grobe Genauigkeit der Näherungslösung gefordert, so wird die Funktion `ode23` empfohlen, da sie weniger Rechenaufwand erfordert. Beide Routinen unterscheiden sich in ihrer programmiertechnischen Handhabung nicht. Deshalb beschreiben wir im folgenden nur `ode23`.

Da die Anwendung der Routinen für retardierte analog zu denen für gewöhnliche Differentialgleichungen verläuft, werden wir nur kurz auf ODE's eingehen.

Matlab löst den folgenden Typ von Anfangswertaufgaben:

$$\begin{array}{rcl} \dot{y}_1 &=& f_1(t, y_1, \ldots, y_n) \quad , \quad y_1(t_0) = y_1^{(0)} \\ \vdots & & \vdots \\ \dot{y}_n &=& f_n(t, y_1, \ldots, y_n) \quad , \quad y_n(t_0) = y_n^{(0)} \end{array}$$

Die Matlab-Funktion `ode23` hat die Syntax

```
sol=ode23(odefun,tspan,y0)
```

Dabei ist `odefun` eine vektorwertige Funktion für die rechte Seite der Differentialgleichung. `tspan` gibt das Zeitintervall $[t_0, t_{end}]$ an, in dem eine Näherungslösung gesucht wird. Der Vektor `y0` enthält die Anfangswerte $(y_1^{(0)}, \ldots, y_n^{(0)})$.

2.1 Beispiel

Als Beispiel lösen wir nun die Anfangswertaufgabe

$$\begin{array}{rclcrcl} \dot{y}_1 &=& -y_2 - y_1(y_1^2+y_2^2-1) &,& y_1(0) &=& 1 \\ \dot{y}_2 &=& y_1 - y_2(y_1^2+y_2^2-1) &,& y_2(0) &=& 0 \end{array} \quad . \tag{1}$$

Die exakte Lösung dieser Anfangswertaufgabe ist $y(x) = (y_1(x), y_2(x)) = (\cos(x), \sin(x))$. Zuerst erstellen wir eine Matlab-Funktion für die Differentialgleichung, welche in die Datei `odefun.m` gespeichert wird:

```
function dydx = odefun(x,y)
dydx=[-y(2)-y(1)*(y(1)*y(1)+y(2)*y(2)-1);y(1)-y(2)*(y(1)*y(1)+y(2)*y(2)-1)];
```

Nun können wir die Lösungsroutine anwenden:

```
sol=ode23(@odefun,[0,10],[1 0]),
```

wobei das @-Zeichen Matlab mitteilt, dass die Funktion in einer Datei steht. Das Output-Argument `sol` besteht aus mehreren Feldern:

- `sol.x` Von der Routine gewählte Gitterpunkte
- `sol.y` Approximation von $y(x)$ an den Gitterpunkten. Dabei ist `sol.y` eine Matrix, die in Zeile k die berechneten Näherunswerte für $y_k(x_i), i = 0, \ldots$ enthält.

Das Zeitbild erhalten wir durch

```
plot(sol.x,sol.y);
```

Alternativ kann man die Komponenten von y auch einzeln zeichnen:

```
plot(sol.x,sol.y(1,:),'r');
hold on;
plot(sol.x,sol.y(2,:),'g');
```

Der Befehl `hold on;` ermöglicht es, eine weitere Kurve in ein bestehendes Bild einzuzeichnen. Die Optionen `'r'` und `'g'` stehen für die Farben rot und grün der einzuzeichnenden Kurven. Das Phasenbild erhalten wir indem wir die erste Zeile von `sol.y` gegen die zweite auftragen

```
plot(sol.y(1,:),sol.y(2,:));
```

2.2 Optionen

Daneben können weitere Optionen gesetzt werden, um etwa die Genauigkeit oder die Ausgabe der Näherungslösung zu verändern. Welche Optionen es insgesamt gibt und welche Voreinstellungen verwendet werden, erfahren Sie durch Eingabe von `odeset`. Diese Funktion wird auch verwendet, um gewisse Parameter zu setzen und hat dann die Form

```
options = odeset('name1',wert1,'name2',wert2,...)
```

Darüber hinaus lassen sich die Art der Ausgabe steuern, parameterabhängige Gleichungen studieren oder auch die Lösung zu festen Zeitpunkten studieren. Auf diese Optionen wollen wir aber erst im nächsten Abschnitt eingehen.

3 Lösen von DDE's mit Matlab

Zum Lösen von DDE's vom Typ

$$\dot{y}(t) = f(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_k))$$

mit zugehörigen Anfangsdaten $\phi(t)$ bietet Matlab die Funktion `dde23` an, die folgende Syntax besitzt:

```
sol = dde23(ddefun,lags,history,tspan,(options))
```

Die Argumente sind dabei

- `ddefun` Eine Funktion, die die rechte Seite der DDE angibt. Sie muss von der Form

  ```
  dydt=ddefun(t,y,Z)
  ```

  sein, wobei t die unabhängige Variable, y der Spaltenvektor der abhängigen Variable ist und `Z(:,j)` ist $y(t-\tau_j)$ für $\tau_j =$`lags(j)`.

- `lags` Ein Vektor, der die Zeitverzögerungen $\tau_1, \ldots, \tau_k$ enthält.

- `history` Eine Funktion von t, die die Anfangsdaten enthält. Sie muss von der Form

  ```
  S=history(t)
  ```

  sein, wobei `S` ein Spaltenvektor ist. Alternativ kann `history` auch ein konstanter Vektor sein.

- `tspan` Das Integrationsintervall als zwei-elementiger Vektor `[t0,tf]` mit $t0 < tf$

Das Output-Argument `sol` besteht aus mehreren Feldern, analog zu der Routine `ode23`.

3.1 Beispiel

Das folgende Beispiel illustriert, wie man DDE's mit Hilfe von matlab löst: Wir betrachten das Beispiel

$$\begin{aligned} y_1'(t) &= y_1(t-1) \\ y_2'(t) &= y_1(t-1) + y_2(t-0.2) \\ y_3'(t) &= y_2(t) \end{aligned}$$

mit konstanten Anfangsdaten $y_i(t) = 1$ für $-1 \leq t \leq 0$ und wollen es auf dem Intervall $[0, 2]$ lösen. Dazu gehen wir wie folgt vor:

(i) **Schreibe das Problem als System erster Ordnung** Im vorliegenden Fall unnötig

(ii) **Definiere die Zeitverzögerungen** Hier `lags=[1,0.2]`

(iii) **Definiere die rechte Seite** Dazu erstellen wir eine Matlab-Funktion, die in die Datei `ddefun.m` gespeichert wird:

```
function dydt = ddefun(t,y,Z)
ylag1 = Z(:,1);
ylag2 = Z(:,2);
dydt = [ ylag1(1)
         ylag1(1) + ylag2(2)
         y(2) ];
```

(iv) **Definiere die Anfangsfunktion** In diesem Fall ist die Anfangsfunktion ein konstanter Vektor, wir schreiben

```
function S = ddehist(t)
S = ones(3,1);
```

in eine Datei mit Namen `ddehist.m`. Dabei bezeichnet `ones(3,1)` den Spaltenvektor der nur Einsen als Einträge besitzt.

(v) **Anwendung der Routine** Mit Hilfe der vorher definierten Funktionen lässt sich nun mit `dde23` die Lösung numerisch berechnen

```
sol = dde23(@ddefun,lags,@ddehist,[0,2]);
```

(vi) **Ansicht der Resultate** Mit `sol.x` bzw `sol.y` kann man sich die entsprechenden Vektoren auf dem Bildschirm ansehen. Eine Visualisierung könnte wie folgt aussehen

```
plot(sol.x,sol.y);
title('Mein erster DDE-Plot :-))');
xlabel('t');
ylabel('y(t)');
legend('y_1','y_2','y_3',2)
```

wobei die 2 angibt, dass die Legende links oben erscheinen soll.

3.1.1 Aufgabe

- Führen Sie obige Schritte durch.
- Tragen Sie nun die Ableitungen $y_i'(t)$ gegen t auf. Lässt sich erkennen, ob die y_i glatt sind ?
- Lösen Sie die DDE wenn die Anfangsdaten durch

$$y_1(t) = \exp(-t), y_2(t) = \exp(-t), y_3(t) = \exp(-t) \tag{2}$$

gegeben sind.

3.2 Optionen

Manchmal ist es nötig, weitere Optionen zu setzen, um etwa die Genauigkeit oder die Ausgabe der Näherungslösung zu verändern. Welche Optionen es insgesamt gibt und welche Voreinstellungen verwendet werden, erfahren Sie durch Eingabe von `help DDESET`. Diese Funktion wird auch verwendet, um gewisse Parameter zu setzen und hat dann die Form

```
options = ddeset('NAME1',VALUE1,'NAME2',VALUE2,...)
```

Eine Option ist beispielsweise die Veränderung der relativen Fehlertoleranz, nach Eingabe von `help DDESET` finden wir

`RelTol` - Relative error tolerance [positive scalar 1e-3] This scalar applies to all components of the solution vector, and defaults to 1e-3 (0.1 accuracy) in all solvers. The estimated error in each integration step satisfies e(i) <= max(RelTol*abs(y(i)),AbsTol(i))

Der Befehl

```
options = ddeset('RelTol',1e-4);
```

senkt den voreingestellten Wert für `RelTol` auf 1e-4. Beim Aufruf von `dde23` muss dann das Argument `options` mit übergeben werden.

Manchmal hängt die DDE noch von Parametern ab. Diese können direkt an `dde23` übergeben werden: Besitzt die Differentialgleichung selbst Parameter, so werden diese als weitere Parameter an die `dde23`-Routine übergeben:

```
sol = dde23(@ddefun,lags,@ddehist,[0,2],options,lambda);
```

Möchten Sie keine zusätzlichen Optionen benutzen, so müssen anstelle von `options` die Platzhalter `[]` stehen. Die Parameter müssen auch bei der Funktionsdefinition auftreten, beispielsweise

```
function dydt = ddefun(t,y,Z,lambda)
...
```

und vor dem Aufruf gesetzt sein. Auch wenn die Anfangsdaten nicht von dem Parameter abhängen, muss dieser trotzdem übergeben werdeb. So wird etwa der Parametervektor $(1, 0, -2, 7) \in \mathbb{R}^4$ durch

```
lambda = [1;0;-2;7]
```

definiert.

3.2.1 Aufgabe

Wir betrachten die paramterabhängige DDE

$$y'(t) = -\lambda y(t-1)(1+y(t)) \tag{3}$$

mit Anfangsdaten $y(t) = t$ für $t \in [0, 1]$ für verschiedene Werte von λ.

- Schreiben Sie die benötigten Funktionen in Dateien `ddefun2.m` sowie `ddehist2.m`.
- Lösen Sie die DDE näherunsgweise mit Hilfe von `dde23` für $\lambda = 1.5, 2$ für $t \in [0, 20]$ und speichern Sie die Lösung in `sol1, sol2, sol3`. Verwenden Sie die voreingestellten Optionen dazu.
- Plotten Sie das Ergebnis mit folgendem Befehl:

  ```
  plot(sol1.x,sol1.y,sol2.x,sol2.y)
  legend('\lambda=1.5','\lambda=2.0')
  ```

- Berechnen Sie nun die Lösung für $\lambda = 2.5$ und plotten diese. Was ist passiert ? Vergleichen Sie dazu `sol1.x` und `sol3.x`.
- Setzen Sie nun den relativen Fehler auf 1e-6 und den absoluten Fehler auf 1e-10 und berechnen abermals die Lösung für $\lambda = 2.5$ und plotten diese gemeinsam mit denjenigen für $\lambda = 1.5$ und $\lambda = 2$.

3.3 Auswertung der Lösung zu bestimmten Zeiten

Die Routine `dde23` berechnet eine Näherungslösung der DDE und speichert alle Informationen, die nötig sind um sie an beliebigen Punkten auszuwerten in `sol`. Dies geschieht mittels der Funktion `ddeval` in folgender Form. Ist `t` ein Vektor, der die Zeiten t_i enthält, zu denen `sol` ausgewertet werden soll, dann leistet

```
s=ddeval(sol,t)
```

das Gewünschte. Beispielsweise erhält man mit `ddeval(sol,1)` den Funktionswert der Lösung zur Zeit $t = 1$

3.3.1 Aufgabe

Wir betrachten folgende skalare DDE, die chaotisches Verhalten aufweist:

$$y'(t) = \frac{2y(t-2)}{1+y(t-2)^{9.65}} - y(t)$$

mit $y(t) = 0.5$ für $t \in [-1, 0]$.

- Lösen Sie die DDE mit `dde23` für $0 \leq t \leq 100$.
- Berechnen Sie den Wert der Lösung zur Zeit $t = 1$.
- Nun wollen wir $y(t)$ gegen $y(t-2)$ in ein Diagramm zeichnen, was man oft anstelle eines Phasendiagramms tut. Dazu benötigen wir Informationen über $y(t-2)$. Definieren wir zunächst die Zeiten an denen wir die Lösung auswerten wollen:

  ```
  t=linspace(2,100,1000);
  ```

  Informieren Sie sich am besten mit `help linspace` über die Wirkungsweise der Funktion. Nun sind wir in der Lage das $(y(t), y(t-2))$-Diagramm zu zeichnen:

  ```
  y=ddeval(sol,t);
  ylag=ddeval(sol,t-2);
  plot(y,ylag);
  ```

3.4 Sonstiges

An dieser Stelle wollen wir darauf hinweisen, dass es noch verschiedene weitere Optionen gibt, auf die wie hier nicht eingehen. Beispielsweise ist `dde23` in der Lage DDE's mit Unstetigkeiten in der Anfangsfunktion oder in der rechten Seite zu lösen. Oft ist man auch interessiert, den Integrationspozess zu stoppen wenn ein bestimmtes Ereignis eintritt, beispielsweise die Funktion wird maximal. Ein solches Ereignis kann in der Form

$$g(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_k)) = 0$$

geschrieben werden und mit Hilfe der `events`-Option implementiert werden.

Literatur

[1] MATLAB MANUAL

[2] Solving DDEs with MATLAB,
`http://www.radford.edu/ thompson/webddes/index.html`

[3] Initial Value Problems for DDEs,
`http://www.eleceng.ohio-state.edu/matlab/techdoc/math_anal/diffeq10.html#diffeq_dde`