Das Eulerverfahren zum numerischen Lösen von DGLen

Thomas Wassong

FB17 Mathematik
Universität Kassel

06.05.2008

Numerische Verfahren zur Berechnung von Differentialgleichungen

Das Eulerverfahren: Programmierung

Eine Integralgleichung

Jedes Anfangswertproblem lässt sich in eine Integralgleichung umwandeln. Das AWP

$$y'(t) = f(t, y),\ y(t_0) = y_0$$

lautet dann

$$y(t_{end}) = C + \int_{t_0}^{t_{end}} f(t, y)dt.$$

Hierbei ist natürlich vorausgesetzt, dass $f(t, y)$ integrierbar ist. Die Konstante C wird durch den Anfangswert bestimmt, so dass $y(t_0) = y_0$ stimmt.

Berechnen von Flächen

Durch die Umwandlung in eine Integralgleichung verlagert sich das Problem beim Lösen von AWPen auf die Berechnung von Integralen.

Welche Möglichkeiten zur Berechnung von Integralen kennt Ihr und wie lassen sich diese auf unser Problem anwenden?

Berechnen von Flächen

Durch die Umwandlung in eine Integralgleichung verlagert sich das Problem beim Lösen von AWPen auf die Berechnung von Integralen.

Welche Möglichkeiten zur Berechnung von Integralen kennt Ihr und wie lassen sich diese auf unser Problem anwenden?

Durch die Nutzung der Riemannschen Summen ergibt sich folgende Berechnung für die Stützstellen bei n Teilschritten:

$$t_i := t_0 + i * h, \text{ mit } h = \frac{t_{end} - t_0}{n}, \ i = 1..n$$

und

$$y_i := y_{i-1} + h * f(t_{i-1}, y_{i-1}) \text{ mit } h = \frac{t_{end} - t_0}{n}, \ i = 1..n.$$

Diese Berechnungsmethode nennt man das explizite Eulerverfahren zur Berechnung von AWPen.

Eine alternative Interpretation des Eulerverfahrens

Für die bekannten Riemannschen Summen gibt es eine alternative Begründung in Bezug auf AWPen und ihren Richtungsfeldern.

Aufgabe: Überlegt Euch anhand des Richtungsfeldes der DGL $y'(t) = sin(t^2 + y^2)$ und dem Anfangswert $y(0) = 1$ eine mögliche Lösung. Welche Eigenschaften hat diese Lösung und wie können wir sie nutzen?

Eine alternative Interpretation des Eulerverfahrens

Für die bekannten Riemannschen Summen gibt es eine alternative Begründung in Bezug auf AWPen und ihren Richtungsfeldern.

Aufgabe: Überlegt Euch anhand des Richtungsfeldes der DGL $y'(t) = sin(t^2 + y^2)$ und dem Anfangswert $y(0) = 1$ eine mögliche Lösung. Welche Eigenschaften hat diese Lösung und wie können wir sie nutzen?

Durch das Richtungsfeld wird die Steigung in jedem Punkt vorgegeben. Ausgehend von dem Startwert ist die Richtung und damit die Funktion vorgegeben. Diese Eigenschaft werden wir ausnutzen und der Steigung im jeweiligen Punkt folgen, um dann einen neuen Punkt zu erreichen.

Alternative Verfahren

Zur Berechnung der Integrale gibt es einige alternative Verfahren, die bei gleicher Schrittweite h genauer sind.

- implizites Eulerverfahren: Trapezregel

Alternative Verfahren

Zur Berechnung der Integrale gibt es einige alternative Verfahren, die bei gleicher Schrittweite h genauer sind.

- Heun-Verfahren: Trapezregel mit explizitem Eulerverfahren zur Berechnung von y_i

Alternative Verfahren

Zur Berechnung der Integrale gibt es einige alternative Verfahren, die bei gleicher Schrittweite h genauer sind.

- klassisches Runge-Kutta-Verfahren: gewichtetes Mittel aus vier Steigunge

Numerische Verfahren in MuPAD

In MuPAD sind entsprechende Verfahren zur numerisch-grafischen Lösung von AWPen implementiert. Die Funktion lautet

`plot::Ode2d`

Die Parameter der Funktion sind etwas umfangreich:

1. Funktion der DGL
2. Stützstellen
3. Anfangswert

Zusätzlich können die verschiedenen Verfahren zur Berechnung der Lösungen angegeben werden. Dies werden wir hier aber nicht weiter behandeln.

MuPAD - Ein Beispiel

siehe MuPAD!

Aufgaben

Plottet eine Lösung des AWPs

$$y'(t) = \pi * cos(t + y^2(t)) + sin(t^2 + y(t)),\ y(0) = 1$$

zusammen mit dem Richtungsfeld.

Einleitung

Wir wollen das Explizite Eulerverfahren programmieren. Dazu werden wir schrittweise die erforderliche Funktionalität hinzufügen und testen.
Die Umsetzung wird zunächst theoretisch besprochen, bevor Ihr den praktischen Teil durchführt. Die benötigten MuPAD-Konstrukte werden dabei besprochen.
Testet Eure Prozedur jeweils an dem AWP

$$y' = y,\ y(0) = 1 \text{ bis } t_{end} = 3 \text{ und der Schrittweite } n = 10.$$

1. Die Rumpfprozedur

Wir beginnen mit der Rumpfprozedur. Dazu wird die MuPAD-Deklaration einer Prozedur benötigt. Sie lautet:

```
func:=proc(var(s))
begin
end_proc;
```

Ebenso werden wir eine Zählschleife benötigen. Hier lautet die Syntax:

```
for i from start to end do
end_for;
```

Aufgaben

Schreibt ein “Hello World”-Programm für MuPAD. Dafür benötigt Ihr den Befehl print.
Danach erweitert das Programm, dass es mit Hilfe einer for-Schleife (`for i from start to end do end_for;`) die folgende Zahlen.

$$t_i := t_0 + i * h, \text{ mit } h = \frac{t_{end} - t_0}{n}, \ i = 1..n$$

Dabei sollen t_0, t_{end} und die Anzahl der Schritte n als Parameter an die Prozedur übergeben werden.

2. Speichern von berechneten Werte

Um die berechneten Funktionswerte effizient zu speichern, sollten Listen verwendet werden. Dazu im folgenden die wichtigsten Befehle im Umgang mit Listen:

```
>> a:=[1,2,3,4];
[1, 2, 3, 4]
>> a.[3,4]
[1, 2, 3, 4, 3, 4]
>> append(a,3)
[1, 2, 3, 4, 3]
>>a[2],a[4]
2, 4
```

Aufgaben

Passt die bisherigen Prozedur an, in dem die t-Werte in einer Liste gespeichert werden.
Erweitert die Prozedur um die Berechnung der y-Werte. Hier nochmal die Formel:

$$y_i := y_{i-1} + h * f(t_{i-1}, y_{i-1}) \text{ mit } h = \frac{t_{end} - t_0}{n},\ i = 1..n.$$

Dabei müssen die Listen vor der for-Schleife mit `t_liste:=[t_0]` bzw. `y_liste:=[y_0]` initialisiert werden.
Vergesst nicht, die Integral-Funktion und den Anfangswert in die Parameterliste mit aufzunehmen.

3. Grafische Darstellung der Lösung

Nun haben wir das Wichtigste schon geschafft, es fehlt nur noch die visuelle Darstellung. Dazu benötigen wir eine Möglichkeit, die berechneten Punkte miteinander zu verbinden. Dazu zeichnen wir mit der Funktion `plot::Line2d` nacheinander die Verbindungslinien zwischen den berechneten Punkten.Die Syntax lautet

```
plot::Line2d([x_0,y_0],[x_1,y_1]).
```

Zur Verdeutlichung sollte die Farbe der Linie angepasst werden (`LineColor`) sowie die Dicke (`LineWidth`).

Aufgaben

Erweitert Eure Funktionen um die Möglichkeit, die gefunden Punkte, verbunden durch Linien, grafisch darzustellen.
Dabei sollte der Rückgabewert (`return()`) eine Liste der grafischen Objekte sein, die dann vom aufrufendem Nutzer geplottet werden können.

4. Plotten des Richtungsfeldes

Um die Lösung überprüfen zu können, sollte auch das Richtungsfeld geplottet werden (`plot::VectorField2d`). Das grafische Objekt hängt Ihr an die Rückgabeliste an.

5. Plotten der Vergleichslösung

Zum Abschluß kann noch die Vergleichslösung mit `plot::Ode2d` geplottet werden. Auch dieses Objekt wird an die Rückgabeliste angehängt.

6. Anwendung

Betrachtet einmal das Ergebnis der DGL

$$y' = 0.8y - 0.02y^2,\ y(0) = 0.01 \text{ mit } t_{end} = 20.$$

Variert dabei die Anzahl der Stützstellen.

Ende

Ich bedanke mich für Eure Aufmerksamkeit!