Numerische Integration

Oswald Berthold

January 18, 2008

Contents

WS0708 Roehrensimulation[1]

1 Motivation

Warum Integrieren: Lösung einer Differentialgleichung als Integral.

> Eine Gleichung in der neben der unabhängigen Variablen x und einer *gesuchten* Funktion $y = y(x)$ auch deren Ableitungen bis zur Ordnung n auftreten, heisst gewöhnliche Differentialgleichung n-ter Ordnung. Ist x_0 aus dem Definitionsbereich von y , so heissen Werte $y(x_0), y'(x_0), ..., y^{(n-1)}(x_0)$ Anfangsbedingungen, die Differenzialgleichung zusammen mit ihren Angangsbedingungen wird Anfangswertproblem genannt. (hartmann S.344)

Geschlossene Lösungen: nur in wenigen speziellen Fällen möglich.
Daher: Numerisches Integrieren

2 Numerische Integration

2.1 Riemannsche Summen

Ansatz: Teilen Integrations-Intervall [a,b] in n gleich grosse Teile, dann ist:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)h_n = \sum_{i=1}^{n} f_i h_n$$

Der Fehler ist beschränkt durch:

$$|\int_a^b f(x) - \sum_{i=1}^{n} f_i h_n| \leq \frac{b-a}{2} \cdot h_n \cdot \max_{x\in[a,b]} |f'(x)|$$

Konvergiert linear mit der Schrittweite.

2.2 Trapezregel, Keplersche Fassregel, Simpson'sche Regel

Bei Anwendung der Trapezregel werden die Punkte $f(x_n), f(x_{n+1})$ mit einer Geraden verbunden, das Integral ergibt sich zu:

$$T_n = h[\frac{1}{2}(f_0 + f_n)] + \sum_{i=1}^{n-1} f_i$$

Konvergiert quadratisch in h_n.
Fassregel: quadratischer Spline durch $f(a), f(\frac{a+b}{2}), f(b)$, das für jedes Teilintervall ist die Simpson'sche Regel.
Damit lassen sich Polynome 3. Grades exakt integrieren.
Fehler und Schrittweite: im Prinzip beliebige Genauigkeit durch beliebig kleine Schrittweiten in Tradeoff mit Rechenzeit und Rechengenauigkeit.

[1] WS0708 Roehrensimulation.pdf

3 Numerische Lösung von Differentialgleichungen

Betrachte nur ODE
Ansonsten: ODE, DAE, DDE, PDE.
(DAE: Differential Algebraic Equation) (DDE: Delay Differential Equation) (PDE: partielle Differentialgleichung: mehrere unabhängige Variablen)
Schritt vom Integrieren einer bekannten Funktion zur Integration als Lösung der DGL:
Gleichung der Form

$$\dot{x} = f(t, x) \quad x(t_0) = x_0$$

ist Anfangswertproblem und das Integral ist die Lösung, nämlich die gesuchte Funktion x.

3.1 ODE

Gewöhnliche Differentialgleichung, es treten nur Ableitungen nach einer Variablen auf. Siehe Definition eingangs.
Anfangswert- / Randwert-Probleme, Anfangswertproblem, S.60
Gewöhnliche DGLs n-ter Ordnung lassen sich immer auf ein System 1. Ordnung von n Gleichungen reduzieren (Angermann S.60)

$$\tilde{x}^{(n)} = f\left(t, \tilde{x}, \dot{\tilde{x}}, ..., \tilde{x}^{n-1}\right)$$

kann in

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &... \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= f(t, x_1, x_2, ..., x_n) \end{aligned}$$

umgeschrieben werden, das ist das Gleichungssytem mit n Gleichungen.

3.1.1 Lösungverfahren / Solver (in Matlab)

Problem:

$$\dot{x}(t) = F(x(t))$$

oder in anderer Schreibweise

$$\dot{x} = F(x, t)$$

Mathematische Bedingungen für die Existenz einer Lösung: Lipschitz-Stetigkeit, d.h. die Funktion macht zu grossen Sprünge.

- Konstante Schrittweite (Fixed Stepsize)
  - Euler: ode1
  - Heun: ode2
  - Bogacki-Shampine: ode3

  - Runge-Kutta: der Standard-Solver und meist erste Wahl: ode4: , ... S.60, matlab doku, automatische oder 'manuelle' Schrittweitenwahl
  - Dormand-Prince: ode5

- Variable Schrittweite (Variable Stepsize)
  - Dormand-Prince: Matlab default ode45 Solver. Berechnet Lösungen 4-ter und 5-ter Ordnung für die Fehlerabschätzung. "embedded formula nach Fehlberg
  - Bogacki-Shampine (ode23)
  - Adams:
  - Stiff/NDF: ode15s Standard Solver für steife Probleme, variable Ordnung
  - Stiff/Mod. Rosenbrock

NDF: numerical differentiation formulas (ode15s) BDF: backwards differentiation formulas
Die unterschiedlichen Verfahrensordnungen geben einen Zusammenhang an zwischen Genauigkeit, Schrittweite und Berechnungsaufwand.

- Euler: Ordnung 1, Verdoppelung der Genauigkeit durch Halbierung der Schrittweite heisst Verdoppelung des Aufwands.
- RK: Ordnung 4, d.h. Fehler auf (1/2)^4 = 1/16 reduziert durch Halbierung der Schrittweite

Abschätzen des Fehlers: Ausführen für verschiedene Schrittweiten oder verschiedene Methoden.

3.1.2 Weitere Aspekte

Variable Order Methods
Continous/Discrete Solver: Kontinuierliche Zustände oder diskrete Zustände.

Implizite / Explizite Methoden Implizite Verfahren beinhalten das Lösen einer Gleichung. Ein Integrationsschritt ist im expliziten Verfahren aber weniger aufwendig, daher sollte ein implizites Verfahren nur angewendet werden, wenn grössere Schrittweiten gleiche Genauigkeit liefern.

Steifheit / Stiffness Lösung durchlaeuft schnelle Änderungen und langsame Änderungen (z.B. Schaltvorgänge): ode15s
Mathematisch: Realteile der Eigenwerte der Jacobi-Matrix sind negativ.

Verfahren mit variabler Schrittweite Finden der geeigneten Schrittweite durch Betrachtung des Fehlers aus unterschiedlichen Schrittweiten/Methoden. Aufwand zur Fehlerermittlung muss immer durch Vergrösserung der Schrittweite gewonnen werden.

3.1.3 Unterschiede in Matlab und Octave

Matlab

Octave

- lsode p.199
- solver(... liefert in Matlab ein struct, braucht dann noch deval(), in octave direkt die Werte an den Stützstellen.
- reihefolge der parameter

3.2 Fazit

Langzeitverhalten: Abklingen, Explodieren, asymptotische Periodizität, Chaotisches Verhalten
ODE-Solver sind effektive Werkzeuge, können aber nicht immer blind angewendet werden. Jegliches Wissen über Eigenschaften der Lösung hilft.
Was Simulink leistet: Übersetzung der Problemformulierung von Dataflow-Beschreibung nach System von n Differentialgleichungen.
Siehe Waveguide-Modelle für Vereinfachungen.

4 Beispiele

4.1 Bsp 1: Pendel mit Herleitung

Ein Standardbeispiel für die numerische Lösung von DGLs ist das (Feder-)Pendel.
Definieren die folgenden Variablen

- x = Position der Masse
- v = x' = Geschwindigkeit der Masse
- m = Masse
- R = Restlänge der Feder
- k = Federsteifigkeit
- b = Reibungsdämpfung

Die Feder erzeugt eine zu ihrer Auslenkung proportionale Kraft in die Gegenrichtung zur Auslenkung:

$$F_{feder} = -k * Auslenkung$$

Setzen wir das Koordinatensystem dass bei Auslenkung $x = 0$ ist, dann erhalten wir:

$$F_{feder} = -kx$$

Die dämpfende Reibungskraft ist proportional der Geschwindigkeit, also $F_{daempfung} = -bv$, setzen wir das ein:

$$F_{gesamt} = F_{feder} + F_{daempfung} = -kx - bv$$

Kombiniert mit Newtons zweitem Bewegungsgesetz $F = ma$ und der Beschleunigung als 2. Ableitung der Position $a = x''$ erhalten wir die Differentialgleichung

$$mx" = -kx - bv$$

und nach einer Umformung

$$x'' = -\frac{k}{m}x - \frac{b}{m}v$$

Das ist die Bewegungsgleichung der Feder mit Gewicht.
Wir wollen das Problem jetzt numerisch lösen unter Anwendung des Runge-Kutta Verfahrens 4. Ordnung. Dazu müssen wir die DGL in die vom Algorithmus geforderte Form bringen. Die Beschleunigung ist die erste Ableitung der Geschwindigkeit: $x'' = v'$, damit können wir die Gleichung umschreiben als System von 2 Gleichungen:

$$\begin{aligned} x' &= v \\ v' &= -\tfrac{k}{m}x - \tfrac{b}{m}v \end{aligned}$$

Jetzt müssen wir die Anfangswerte zum Zeitpunkt $t = 0$, x_0, v_0 setzen und die Lösung / Simulation kann beginnen.
mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/numerische.integration.m[2]

4.2 Bsp 2: Schwingkreis

Nach Barkhausen Tabelle S.31 haben wir für einen elektrischen LC-Schwingkreis die Bewegungsgleichung:

$$Lq'' + Rq' + \frac{q}{C} = 0$$

Umformuliert für Zuführung zur numerischen Lösung:

$$Lq'' = -Rq' - \tfrac{1}{C}q$$

$$\begin{aligned} q' &= i \\ i' &= -\tfrac{R}{L}i - \tfrac{\frac{1}{C}}{L}q \end{aligned}$$

mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/numerische.integration.m[3]

4.3 Bsp 3: Pendel mit erzwungener Schwingung

Beispiel aus Angermann, S.63, dort angetriebenes Pendel
mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/dgl_pendel.m[4]

4.4 Bsp 4: Saite: unendlich viele Freiheitsgrade

fehlt.

[2] mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/numerische.integration.m
[3] mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/numerische.integration.m
[4] mw/2007_modellierung.der.elektronenroehre/20080111_RefNumInt/dgl_pendel.m

4.5 Bsp 5: Roessler / Lorenz: SuperCollider

- ~/SuperCollider/ode-euler.sc[5]
- ~/SuperCollider/chaos03.sc[6]

5 Refs

- http://en.wikipedia.org/wiki/Ordinary_differential_equations[7]
- http://en.wikipedia.org/wiki/Category:Numerical_differential_equations[8]
- http://en.wikipedia.org/wiki/Numerical_ordinary_differential_equations[9]
- `http://en.wikipedia.org/wiki/Dormand-Prince`
- http://en.wikipedia.org/wiki/Stiff_equation[10]
- http://www.physiome.org/jsim/docs/Solver_ODE_Ref.html[11]
- `http://www1.uni-hamburg.de/W.Wiedl/Skripte/Matlab/ODE/STEIF/`
- `http://www.myphysicslab.com/` ...
- Mathematik für Informatiker, P. Hartmann, Vieweg
- Vorlesungs-Skript Mathematik für Informatiker 3, HU-Berlin, A. Griewank, WS0506
- Barkhausen: Schwingungslehre
- Matlab:
  - Angerman: Kapitel 4 (S.59 - 90), Kapitel 8, S.72, 123
  - Benker: S.255 (423)
  - Beucher: S.118
  - Hoffmann: S.79 (81)

6 XXX

- eigenwert
- schreibweisen vereinheitlichen

[5] /SuperCollider/ode-euler.sc
[6] /SuperCollider/chaos03.sc
[7] http://en.wikipedia.org/wiki/Ordinary_differential_equations
[8] http://en.wikipedia.org/wiki/Category:Numerical_differential_equations
[9] http://en.wikipedia.org/wiki/Numerical_ordinary_differential_equations
[10] http://en.wikipedia.org/wiki/Stiff_equation
[11] http://www.physiome.org/jsim/docs/Solver_ODE_Ref.html