Ereignislokalisierung durch die Henon-Methode

Franz Preyser[1], Bernhard Heinzl[2], Matthias Rößler[3], Felix Breitenecker[1]
[1]Institut für Analysis und Scientific Computing, TU Wien
[2]Institut für Rechnergestützte Automation, TU Wien
[2]dwh GmbH, Simulation Services, Wien
Franz.Preyser@tuwien.ac.at

Ereignislokalisierung spielt im Bereich der Simulation mathematischer Modelle in Form von gewöhnlichen Differentialgleichungssystemen eine wichtige Rolle. Bei den dabei üblicherweise verwendeten Verfahren(z.B. Regula-Falsi), handelt es sich ausschließlich um iterative Verfahren. Im Folgenden soll ein nicht iteratives Verfahren namens "Henon" vorgestellt werden, welches in Matlab implementiert wurde. Nachdem kurz auf die Grenzen der Henon-Methode eingegangen wird, werden abschließend Resultate eines Laufzeitvergleichs mit dem Regula-Falsi-Verfahren präsentiert.

1 Einleitung

Beim üblichen numerischen Lösen von gewöhnlichen Differenzialgleichungssystemen liefert das numerische Lösungsverfahren(ODE-Solver) Lösungswerte zu konkreten Werten der unabhängigen Varaible(meistens die Zeit). Diese Werte haben entweder einen festen Abstand zueinander(fix-step-solver), oder die Abstände sind variabel(variable-step-solver). In manchen Fällen ist es jedoch wünschenswert, Lösungswerte bei ganz bestimmten Systemzuständen zu erhalten, da z.B. ab Erreichen dieses Zustandes, das verwendete Modell nicht mehr gültig ist und daher etwaige Modellparamter oder ganze Gleichungen geändert werden müssen.

1.1 Ein Beispiel

Ein Beispiel liefert eine sehr einfache Version des wohl bekannten „Bouncing Ball". Hierbei wird die Postion eines hüpfenden Balles berechnet, indem während der Flugphasen einfach die Bewegungsgleichung(freier Fall)

$$x(t) = x_0 + v_0 \cdot t - \frac{g}{2} \cdot t^2 \tag{1}$$

gelöst wird. Diese Gleichung gilt aber nur, solange der Ball nicht am Boden aufschlägt. D.h. bei Erreichen des Systemzustandes $x(t) = 0$ muss das numerische Lösungsverfahren abgebrochen werden um entweder das Modell zu ändern, oder, wie hier möglich, den Systemzustand(Reflexion des Balles: $v(t) \mapsto -v(t)$). Es muss also das numerische Lösungsverfahren dazu gebracht werden, einen Lösungswert genau beim Aufprallzeitpunkt zu berechnen und danach abzubrechen. In weiterer Folge soll die unabhängige Variable stets als Zeit t bezeichnet werden. Desweiteren verwenden wir fett gedruckte Variablen für vektorwertige Größen(z.B. Systemzustand $\mathbf{x}$).

1.2 Übliche Verfahren

Die übliche Herangehensweise ist nun, eine sogenannte Ereignisfunktion

$$S : \Omega \to \mathbb{R} \tag{2}$$

zu definieren. Diese Funktion bildet den Zustandsraum Ω stetig in die reellen Zahlen ab und ihre Nulldurchgänge kennzeichnen genau die Ereigniszeitpunkte. Im oben genannten Beispiel wäre das gesuchte Ereignis, das Erreichen der Ballhöhe 0 und die Ereignisfunktion deshalb:

$$S(x) = x \tag{3}$$

Das numerische Lösungsverfahren überprüft nach jeder Lösungswertberechnung, ob sich das Vorzeichen der Ereignisfunktion S geändert hat. Wird ein Vorzeichenwechsel detektiert, so wird ein Algorithmus zur Nullstellenlokalisierung angeworfen. Übliche Verfahren sind z.B. das Newton-, das Sekanten-, das Bisektions- oder das Regula-Falsi- Verfahren. Dabei handelt es sich ausnahmslos um iterative Verfahren. Das im Folgenden vorgestellte Henon-Verfahren, benannt nach dem Autor des Papers [1], aus dem die Idee zum Algorithmus stammt, soll hier einen anderen Weg gehen.

1.3 Henon

Das Henon-Verfahren kann als eine Erweiterung des Newton-Verfahrens gesehen werden und stellt deshalb auch dieselben Ansprüche an die Funktion, deren Nullstelle lokalisiert werden soll. In unserem Fall die Ereignisfunktion. Vernachlässigt man den numerischen Fehler und geht also davon aus, dass die für den Systemzustand $\mathbf{x}$ berechneten Lösungswerte $\mathbf{x}_n$ gleich den Werten der analytischen Lösung $\mathbf{x}(t_n)$ sind, so kann die Ereignisfunktion auch als Funktion der Zeit gesehen werden:

$$\tau(t) = S(\mathbf{x}(t)) \tag{4}$$

Ist nun diese Funktion τ in dem Intervall $(t_n, t_e + (t_e - t_n))$ stetig und monoton, wobei $sign(\tau(t_n)) \neq sign(\tau(t_{n+1}))$ und $\tau(t_e) = 0$ gelte(t_e ... Ereigniszeitpunkt) und ist zusätzlich die Ableitung $\frac{d\tau}{dt}$ in dem Intervall bekannt, so kann das Newton-Verfahren angewandt werden. Dieses macht nichts anderes, als die Nullstelle $\tilde{t}_e$ der Tangente im Punkt $\tau(t_n)$ zu berechnen und als erste Näherungslösung für die Nullstelle t_e von τ heranzuziehen:

$$\tilde{t}_e = t_n - \left(\frac{d\tau}{dt}(t_n)\right)^{-1} \cdot \tau(t_n) \tag{5}$$

Anschließend wird der Systemzustand zum Zeitpunkt $\tilde{t}_e$ berechnet, um danach erneut die Nullstelle der Tangente im Punkt $\tau(\tilde{t}_e)$ zu ermitteln. Dieser Vorgang wird solange iteriert, bis $\tau(\tilde{t}_e)$ nahe genug bei Null und somit $\tilde{t}_e$ nahe genug bei t_e liegt. Gleichung (5) kann jedoch auch anders interpretiert werden: Da mit den obigen Voraussetzungen $\tau(t)$ im Intervall (t_n, t_e) invertierbar ist und die Inverse im Punkt $\tau(t_n)$ die Steigung $\left(\frac{d\tau}{dt}(t_n)\right)^{-1}$ hat, entspricht (5) genau einem Schritt des expliziten Eulerverfahrens mit Schrittweite $-\tau(t_n)$, angewandt auf das AWP:

$$\begin{cases} \frac{dt}{d\tau}(\tau) & = \left(\frac{d\tau}{dt}(t)\right)^{-1} \\ t(\tau_n) & = t_n \end{cases} \tag{6}$$

Es liegt daher nahe, anstelle der iterativen Anwendung des Newton-/Expliziten Euler-Verfahrens, einmalig ein Verfahren höherer Ordnung anzuwenden, wie z.B. Runge-Kutta. Beim Henon-Verfahren passiert genau das. Eine wesentliche Voraussetzungen für Newton wie auch für Henon ist also, im Unterschied zu allen anderen aufgezählten Methoden, dass die Ableitung der Ereignisfunktion analytisch bekannt und die Ereignisfunktion in einem kleinen Bereich um die zu lokalisierende Nullstelle invertierbar sein muss.

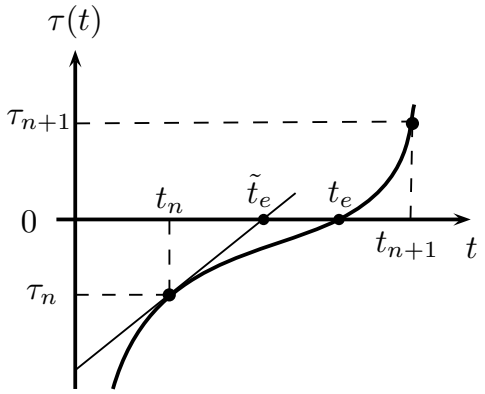

Abbildung 1: Eine Iteration im Newton-Verfahren.

2 Matlab - Implementierung

2.1 Algorithmusaufbau und Funktionsweise

Um die Brauchbarkeit der Henon-Methode zu überprüfen, haben wir sie in Matlab implementiert. Konkret wurde eine Wrapper-Funktion für Matlab ODE-Solver geschrieben, wobei auch der zu verwendende Solver(z.B. ode45) als Parameter übergeben werden kann. Diese Wrapper-Funktion kann somit völlig universell auf jedes Beispiel angewandt werden, bei dem üblicherweise ein Matlab-Solver zum Einsatz kommt und die Voraussetzungen für das Henon-Verfahren erfüllt sind. Es handelt sich dabei also um

Anfangswertprobleme(AWPs) der Bauart:

$$\begin{cases} \frac{d\mathbf{x}}{dt}(t) & = \mathbf{f}(t,\mathbf{x}) \\ \mathbf{x}(t_0) & = \mathbf{x}_0 \end{cases} \tag{7}$$

Zusätzlich zum AWP (7) muss noch eine Ereignisfunktion $S(\mathbf{x})$ angegeben werden. Sowohl zum Lösen von (7), als auch für das Henon-Verfahren wird der mittels Parameter festgelegte Solver verwendet. Es ist sogar möglich AWP (7) und AWP (6) so zusammenzufassen, dass lediglich der Wert einer Variablen K entscheidet, ob beim Lösen des resultierenden AWPs gerade das eigentliche Problem (7) gelöst, oder eine Nullstelle der Ereignisfunktion lokalisiert wird.

$$\begin{cases} \frac{d\mathbf{y}}{d\tau}(\tau) & = \mathbf{f}(t(\tau),\mathbf{y}) \cdot K(\tau,\mathbf{y}) \\ \frac{dt}{d\tau}(\tau) & = K(\tau,\mathbf{y}) \\ \mathbf{y}(\tau_0) & = \mathbf{y}_0 \\ t(\tau_0) & = t_0 \end{cases} \tag{8}$$

(8) zeigt die Gestalt des oben erwähnten zusammengefassten AWPs. Die unabhängige Variable τ nimmt nun je nach Wert von K die Rolle der Zeit t ($K \equiv 1$) oder die Rolle der Ereignisfunktion $(S \circ \mathbf{x})(t)$ ($K = \left(\frac{d\tau}{dt}\right)^{-1}$) an. $\mathbf{y}$ entspricht in jedem Fall dem Systemzustand $\mathbf{x}$, nur einmal als Funktion der Zeit und einmal als Funktion des Ereignisfunktionswertes. Möchte man nun eine zwischen den Zeitpunkten τ_n und τ_{n+1} detektierte Nullstelle lokalisieren, ändert man den Wert von K von 1 auf $\left(\frac{d\tau}{dt}\right)^{-1}$, integriert (8) von $\tau = S(y_n)$ bis $\tau = 0$ und erhält somit den Systemzustand $\mathbf{x}(t_e) = \mathbf{y}(0)$ zum Ereigniszeitpunkt t_e, sowie t_e selbst. Anschließend wird K wieder zurück auf 1 gesetzt und mit der Integration von (8) bei $\tau = t_e$ fortgesetzt.

2.2 Vergleiche

Es wurde die Henon-Implementierung mit dem Regula-Falis-Verfahren verglichen. Um den Vergleich möglichst fair zu gestalten, wurde nicht der Matlab-eigene Lokalisierungsmechanismus verwendet, sondern Regula-Falsi analog zu Henon, als Wrapper-Funktion implementiert. Betrachtet wurde die Performance beim Lösen des gedämpften und ungedämpften Bouncing-Ball Beispiels, unter Berücksichtigung der benötigten Sample und des gemachten Fehlers. Die zu lokalisierenden Ereignisse waren dabei die Nullstellen der Ball-Flugbahn sowie deren Maximumsstellen. Als ODE-Solver wurden die Matlab-Solver *ode23* und *ode45*, als auch ein eigens programmierter fix-step-Solver verwendet, wobei letzterer nach einem expliziten Runge-Kutta-Verfahren 4. Ordnung und 4. Stufe arbeitet. Der fix-step-Solver dient zum gezielten Aufzeigen der Grenzen des Henon-Algorithmus(siehe Abschnitt 2.3).

Beispiel	**Solver**	$\frac{t_{RF}}{t_H}$	$\frac{t_{ML}}{t_H}$
BB	*ode23*	8.75	0.19
d BB	*ode23*	3.13	0.14
BB M	*ode23*	0.60	0.11
d BB M	*ode23*	1.06	0.10
BB	*ode45*	4.76	0.35
d BB	*ode45*	3.13	0.14
BB M	*ode45*	4.43	0.35
d BB M	*ode45*	2.98	0.25

Tabelle 1: Ergebnis Laufzeitvergleich:
t_H ...Laufzeit der Henon-Implementierung
t_{RF} ... Laufzeit der Regula-Falsi-Implementierung
t_{ML} ...Laufzeit des Matlab-Mechanismus
BB ... Bouncing Ball
BB M ...Bouncing Ball mit Maximumslokalisierung
d BB ... gedämpfter Bouncing Ball
d BB M ...ged. Bouncing Ball mit Maximumslokalisierung

Tabelle 1 zeigt die Laufzeit-Resultate der Bouncing-Ball Simulationen. Die Zahlen in den letzten beiden Spalten stellen dabei nicht die konkreten Laufzeiten dar, sondern das Verhältnis von Laufzeit des Regula-Falsi- bzw. des MATLAB-Verfahrens zu Laufzeit des Henon-Verfahrens. Diverse genauigkeitsbestimmende Parameter der Henon- und der Regula-Falsi-Implementierung wurden dabei so gewählt, dass der resultierende Fehler in etwa dem des Matlab-Mechanismus entspricht. Es ist zu erkennen, dass Henon in den meisten Fällen schneller arbeitet als Regula Falsi. Unter Verwendung von *ode23* benötigt Henon beim gewöhnlichen BB sogar weniger als 1/8 der Laufzeit von Regula-Falsi. Dafür ist beim BB mit Maximum-Lokalisierung Henon langsamer als Regula-Falsi. Allgemein hat sich herausgestellt, dass eine große Schrittweite des zugrundeliegenden Solvers das Laufzeitverhältnis zu Gunsten von Henon beeinflusst. Ein Indiz dafür liefert die Tatsache, dass

Henon unter Verwendung von *ode45* im Schnitt 3-mal so schnell arbeitet wie Regula-Falsi. Mit dem Matlab-internen Ereignislokalisierungsmechanismus kann jedoch keine der beiden selbst programmierten Versionen mithalten. Dies ist wohl auf die "umständliche" Implentierung als *Wrapper-Funktion* zurückzuführen.

2.3 Grenzen

An seine Grenzen stößt das Henon-Verfahren, wenn die Ableitung der Ereignisfunktion im Bereich der zu lokalisierenden Nullstelle sehr klein wird. Da die Ableitung beim Henon-Algorithmus in invertierter Form auftritt, kommt es in diesem Fall zu numerischen Schwierigkeiten, die sich in einem, mitunter massiv erhöhten, lokalen Fehler niederschlagen. Dies ist beispielsweise dann der Fall, wenn die analytische Lösung des ODE-Systems $x(t) = 1 - \varepsilon + sin(t)$ lautet, wobei ε als sehr klein vorausgesetzt wird und sowohl die Nullstellen als auch die Extrema lokalisiert werden sollen. In diesem Fall liegen nämlich die Nullstellen und das Minimum von $x(t)$ sehr nahe beieinander. Abbildung 2 zeigt das Ergebnis einer solchen Simulation mit einem ε-Wert von 10^{-10}, wobei der Bereich der Nulldurchgänge stark vergrößert wurde. Man sieht, dass die Henon-Lösung ab dem lokalisierten Minimum stark von der analytischen Lösung abzuweichen beginnt.

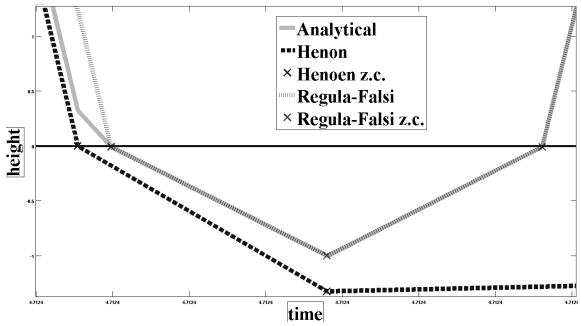

Abbildung 2: Die Lokalisierung der Nullstellen und des Minimums einer verschobenen Sinusfunktion. Die unterste strichlierte Linie zeigt die Lösung des Henon-Algorithmus. Die durchgehend gezeichnete Linie zeigt die abgetastete analytische Lösung und die dritte Linie(nur zu Beginn nicht deckungsgleich mit der analytischen Lösung) zeigt die Regula-Falsi-Lösung.

3 Zusammenfassung und Ausblick

Zusammenfassend ist zu sagen, dass das Henon-Verfahren durchaus Potential hat mit dem Regula-Falsi Verfahren mitzuhalten. Es ergeben sich jedoch eine Vielzahl von genauigkeitsbeeinflussenden Parametern, deren Kalibrierung eine Aufgabe für sich darstellt. Es wäre zudem notwendig mehrere verschiedenartige Beispiel zu betrachten und sich nicht, wie hier, im Wesentlichen auf den Bouncing Ball einzuschränken. Der Matlab-Ereignislokalisierungsmechanismus arbeitet um einiges schneller, was jedoch auf die "umständliche" Implementierung des Henon-Algorithmus als *Wrapper*-Funktion, aber auch auf einige Laufzeit-Optimierungstricks, die wohl beim Matlab-Mechanismus zum Einsatz kommen, zurückzuführen ist. In Sachen "Laufzeit-Optimierungstricks" schlummert jedoch ohne Zweifel auch im Henon-Algorithmus noch ungenutztes Potential. Weitere Untersuchungen wären somit angebracht.

References

[1] M. Hénon: On the numerical Computation of Poincaré Maps, *Physica D: Nonlinear Phenomena, 5(2)*, S. 412–414, 1982.