Übungen aus den Numerische Methoden der Astronomie SS 2010
Programmierbeispiel II - numerische Integration und Differentiation

1 Aufgabe

Ziel dieses Beispiels ist es, die Güte der numerischen Integration und Differentiation der Funktion $f(t) = Sin(t)$ in Abhängigkeit von der Schrittweite h zu vergleichen, sowie die Lösung der Differentialgleichung des eindimensionalen, Harmonischen Oszillators numerisch zu approximieren.

2 Einführung - Wozu Numerik?

Eine der größten Errungenschaften der Analytik, nämlich die Möglichkeit globale Aussagen über das Verhalten von Systemen bezüglich einer Reihe von Anfangsbedingungen und für beliebige Entwicklungszeiträume[1] treffen zu können / wollen, stellt gleichzeitig eines ihrer größten Probleme dar. Wie im Programmierbeispiel I kurz angesprochen existiert z.B. keine elementare, analytische Funktion die das Verhalten von mehr als drei gravitativ gebundenen Körpern für alle Anfangsbedingungen und alle Zeiten repräsentiert. Paradoxerweise lassen sich aber deren Bewegungsabläufe sehr wohl durch Differentialgleichungen mathematisch beschreiben. Worin genau also liegt das Problem? (Viel Spaß beim Nachdenken, Antwort bitte ins Protokoll!)

Da nicht immer analytisch exakte Lösungen von Differentialgleichungen (=DGL) für alle Anfangsbedingungen und alle Zeiten existieren (oder auffindbar sind), besteht ein naheliegender Schritt darin *Näherungen* einzusetzen. Die Computerisation der heutigen Gesellschaft ermöglicht Zugang zu enormen Rechenleistungen, daher stellen Numerik und diskrete Mathematik oft Methoden erster Wahl dar. Der Übergang von einer analytischen zu einer disktreten Betrachtungsweise von Problemen lässt sich am besten anhand von Differenzen- und Differentialquotienten zeigen.

[1] Jener Parameter, der die Entwicklung eines Systems beschreibt (sollte es überhaupt einen geben), ist natürlich keineswegs notwendigerweise mit der Zeit zu identifizieren. Der Anschaulichkeit halber wird jedoch in der Folge von der zeitlichen Entwicklung eines Systems gesprochen.

2.1 Numerische Differentiation

Gegeben sei eine Funktion $f : \mathbb{R} \to \mathbb{R}$, wobei wir $h = t - t_0$ als Schrittweite bezeichnen wollen, mit $\{t, t_0, h\} \in \mathbb{R}$. Kombiniert man den Hauptsatz der Integral und Differentialrechnung mit dem Mittelwertsatz und bildet den Limes für Schrittweite $h \to 0$ so erhält man die Gleichung für den Differentialquotienten (Wikipedia Proof 1, 2009):

$$\lim_{h \to 0} \frac{f(t_0 + h) - f(t_0)}{h} = \dot{f}(t_0)$$

Hat man es mit Zahlen endlicher Präzision zu tun, bringt die Einführung von infinitesimalen Größen oder Limiten gröbere Probleme mit sich. Daher wird schlicht der Limes $h \to 0$ vernachlässigt, und es wird akzeptiert, dass es sich in der Folge um einen Differenzenquotienten, also lediglich um eine Näherung an $\dot{f}(t)$ handelt.

$$\frac{f(t_0 + h) - f(t_0)}{h} \simeq \dot{f}(t_0) \tag{1}$$

oder gemäß der Nomenklatur der Vorlesung:

$$\frac{\Delta_h f(t_0)}{h} \simeq \dot{f}(t_0) \tag{2}$$

Sowohl die Funktionswerte $f(t_0 + h), f(t_0)$ als auch die Schrittweite h sind nun durch diskrete Zahlen repräsentierbar. Gleichungen (1) und (2) stellen somit eine *erste Näherung der Ableitung* $\frac{df(t)}{dt}$ dar, die über Differenzen von Funktionswerten berechnet werden kann.

2.2 Numerische Integration

In Relation (1) ist allerdings auch eine Vorschrift versteckt, die eine numerische Integration ermöglicht. Gegeben seien eine gewöhnliche DGL 1.Ordnung in t, wobei $f, g : \mathbb{R} \to \mathbb{R}$ mit $t \in \mathbb{R}$:

$$\frac{df(t)}{dt} = g(t, f(t)) \tag{3}$$

Um eine Approximation für die Lösungsfunktion $f(t)$ zu erhalten schreiben wir Relation (1) wie folgt um:

$$f(t_0 + h) \simeq f(t_0) + h \cdot \dot{f}(t_0) \tag{4}$$

$$f(t) \simeq f(t_0) + h \cdot \dot{f}(t_0) \tag{5}$$

Die Gleichung (5) liefert in der Tat den (genäherten) Wert der Stammfunktion $f(t)$, also der Lösungsfunktion der DGL (3), vorausgesetzt man kennt die Anfangsbedingungen $f(t_0)$ und $\dot{f}(t_0)$, denn die Schrittweite h ist prinzipiell frei wählbar. Bei genauerer Betrachtung fällt jedoch auf, dass diese Näherung einer linearen Extrapolation vom Zeitpunkt t_0 auf t entspricht. *Wie gut ist die Näherung an die echte Stammfunktion also wirklich?* Um dieser Frage nachzugehen bedienen wir uns der *Taylorreihenentlwicklung* der Funktion $f(t_0 + h)$ um den Punkt t_0 im Parameter h bis zur 3.Ordnung:

$$f(t_0 + h) = f(t_0) + hf'(t_0) + \frac{1}{2}h^2 f''(t_0) + O(h^3) \tag{6}$$

Vergleichen wir dies mit unserer Näherung (4), so stimmen die beiden Gleichungen bis auf die Terme

$$\frac{1}{2}h^2 f''(t_0) + O(h^3)$$

überein. Dies bedeutet, unser Integrationsverfahren (5) gleicht der tatsächlichen Stammfunktion $f(t)$ bis auf Terme $O(h^n), n \geq 2$. Deshalb wird es als Verfahren *erster Ordnung* kategorisiert. Korrekt formuliert lautet unser Integrationsverfahren erster Ordnung:

$$f(t) = f(t_0) + h \cdot \dot{f}(t_0) + O(h^2) \tag{7}$$

Da wir dieses Integrationsschema - wie so vieles in der Mathematik - Leonhard Euler (1707-1783) zu verdanken haben, taucht es in der Literatur meist unter dem Namen *Eulerverfahren* auf.

Wie zuvor erwähnt entspricht das Eulerverfahren im Prinzip einer linearen Extrapolation, es ist also mit Sicherheit nicht der Weisheit letzter Schluss. Tatsächlich hat sich vor allem im letzten Jahrhundert ein ganzer Zweig der Mathematik mit unzähligen Ästen - sprich Verfahren - entwickelt, der sich ausschließlich mit Problemen effizienter numerischer Integration auseinandersetzt. Dabei reichen die Ansätze von einfachen geometrischen Überlegungen (z.B. Mittelpunkts- und Trapezregel) über die Anwendung von Interpolations- (Simpson-Quadratur) und Extrapolationspolynomen (Adams-Bashforth-Moulton Predictor-Corrector, in abstrakter Weise auch der Bulirsch-Stoer Algorithmus), sequentiellen Anstiegsverbesserungen (Runge-Kutta) bis hin zu mathematisch anspruchsvolleren Reihenentwicklungen (Lie-Reihen Integrator), und die Phasenraumgeometrie erhaltenden, "symplektischen" Integratoren.

Allen diesen Verfahren ist jedoch gemein, dass es sich dabei um Näherungen handelt, deren Güte in Form von Polynomen in der Schrittweite h angegeben werden kann. Daher ist die Bedeutung letzterer für die Stabilität der Verfahren und deren "Abwanderungsverhalten" von der tatsächlichen Stammfunktion enorm. Eine Faustregel für explizite[2] Verfahren wie z.B das Eulerverfahren besagt: *Ist die Schrittweite zu groß, dann bin ich meine Lösung los.* Die Gleichung (5) lässt vermuten, dass, falls es sich bei der Stammfunktion nicht zufällig um eine lineare Funktion handelt, zu große Schrittweiten h ein globales "Abwandern" der Näherung von der tatsächlichen Stammfunktion verursachen. Deshalb wird zunächst kein "großer" Schritt gemacht um vom Startpunkt des Systems $(f(t_0))$ auf den Endpunkt $(f(t))$ zu gelangen, sondern stattdessen wird das Gesamtintervall $t - t_0$ in viele kleinere Teilintervalle $t - t_{n-1}, t_{n-1} - t_{n-2}, ..., t_1 - t_0$ aufgespalten über die in "kleinen" Schritten integriert wird (siehe Abb. 1).

Somit ergibt sich für die Gesamtschrittweite $h = \sum_{i=1}^{n} h_i$ wobei
$h_i = t_i - t_{i-1}$, $\{i, n\} \in \mathbb{N}_+$ und $t_n = t$ ist. Durch die Unterteilung in Subintervalle folgt aus Gleichung (7) eine Iterationsvorschrift:

$$f(t_i) = f(t_{i-1}) + h_i \cdot \dot{f}(t_{i-1}) + O_{lokal}(h_i^2) \qquad \text{mit } i = 1, 2, \ldots n \tag{8}$$

[2]Explizite Verfahren berechnen $f(t)$ ausschließlich mit Werten $f(t_{i-1,i-2,\ldots}), \dot{f}(t_{i-1,i-2,\ldots})$. Im Gegensatz dazu verwenden implizite Verfahren noch nicht bekannte Werte $f(t), \dot{f}(t)$. Implizite Verfahren sind weniger anfällig auf Instabilitäten verursacht durch zu große Schrittweiten, allerdings mangelt es ihnen an Effizienz, d.h. sie leiden unter unverhältnismäßigem Rechenzeitverbrauch. Siehe z.B. Vesely (2001)

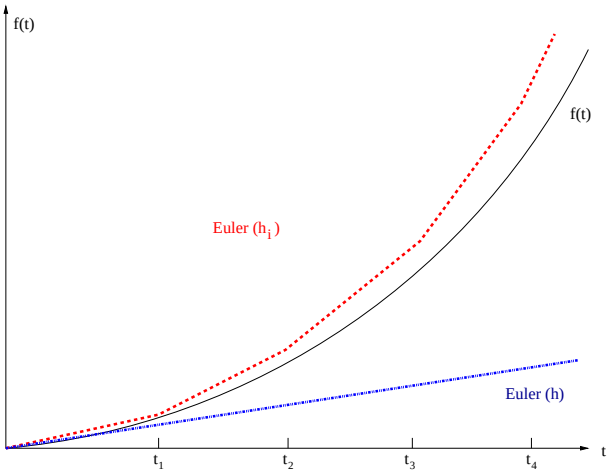

Abbildung 1: Das Eulerverfahren mit einer Schrittweite gleich dem Gesamtintervall ($h = t - t_0$) stellt eine viel schlechtere Approximation an die Stammfunktion $f(t)$ dar als das wiederholt angewendete Eulerverfahren mit kleineren Schrittweiten (h_i)

Bei einfachen Integrationsalgorithmen wird das Gesamtintervall $t - t_0$ in *äquidistante* Subintervalle geteilt, sodass $h_i = h_{i-1} = \bar{h}$, $\forall i$. Diese konstante Schrittweite $\bar{h}$ wird zu Beginn der Integration festgelegt. Unter diesen Voraussetzungen gilt für gesamte Integrationszeit $t - t_0$:

$$\begin{aligned} t - t_0 &= \sum_{i=1}^{n} h_i \\ &= n \cdot \bar{h} \end{aligned} \tag{9}$$

wobei n die Anzahl der durchlaufenen Integrationsschritte darstellt. Je kleiner die konstante Schrittweite $\bar{h}$ desto mehr Integrationsdurchläufe müssen durchgeführt werden, um die gleiche Gesamtintegrationszeit $t - t_0$ zu erzielen.

$f(t_i)$ stellen die Zwischenergebnisse, bzw. Raststationen dar, von denen aus die "Reise" nach $f(t_n) = f(t)$ fortgesetzt wird.[3] Das bedeutet, nach jedem Iterationsschritt bilden die Ergebnisse die neuen Anfangsbedingungen für den nächsten Integrationsschritt.

$$\begin{aligned} f(t_1) &= f(t_0) + \bar{h} \cdot \dot{f}(t_0) \\ f(t_2) &= f(t_1) + \bar{h} \cdot \dot{f}(t_1) \\ &\vdots \\ f(t_n) &= f(t_{n-1}) + \bar{h} \cdot \dot{f}(t_{n-1}) \end{aligned} \tag{10}$$

Hierbei wurde weiters vorausgesetzt, dass die Schrittweite $\bar{h}$ so klein ist, dass

[3] Deshalb spricht man in diesem Fall auch von dem lokalen Fehler der Ordnung O_{lokal}, da die Fehlerabschätzung genau genommen nicht für das gesamte Intervall $t - t_0$, sondern nur für das Teilintervall $t_i - t_{i-1}$ gültig ist.

deren Quadrat $\bar{h}^2$, alle höheren Potenzen, und somit auch alle damit multiplizierten Terme ($O(\bar{h}^2)$) vernachlässigt werden können[4].

Wie aber bestimmt man die Ableitungen zu den jeweiligen Zeiten $\dot{f}(t_{i-1})$? Dies erfolgt naturgemäß durch Einsetzen der Werte von $f(t_{i-1})$ in die Funktion $g(t_{i-1}, f(t_{i-1}))$ der DGL (3). Aus der Iterationsvorschrift (8) und dem Zusammenhang zwischen Schrittweite und Gesamtintegrationszeit (9) lassen sich folgende Fakten ablesen:

1. Je feiner die Unterteilung der Subintervalle, desto kleiner die jeweiligen Schrittweiten h_i. Kleinere h_i bedeuten laut Abschätzung (7) eine bessere Näherung an die Stammfunktion.

2. Je feiner die Unterteilung der Subintervalle, desto mehr Rechenschritte müssen in Kauf genommen werden, um das gleiche Gesamtintegrationsintervall zu erreichen.

Hauptziel bei der Wahl der Schrittweiten h_i, bzw. $\bar{h}$ sollte also ein Gleichgewicht zwischen Rechenaufwand und erzielter Genauigkeit sein. Da man a priori selten die Möglichkeit hat, das Verhalten der Stammfunktion vorherzusagen, kann es passieren, dass sich diese in weiten Bereichen ruhig verhält, in anderen jedoch sehr stark variiert. Verwendet man nun konstante, geringe Schrittweiten, stimmt zwar die Genauigkeit, die Kosten an Rechenzeit werden allerdings enorm hoch. Daher haben sich in der Praxis Verfahren durchgesetzt, in denen eine *automatische Wahl* der lokalen Schrittweiten h_i implementierbar ist (*Schrittweitensteuerung*, siehe z.B. (Cash & Karp, 1990)). Für die folgenden Beispiele soll dennoch das Eulerverfahren mit konstanter Schrittweite $\bar{h}$ verwendet werden.

3 Aufgaben im Detail

3.1 Numerische Differentiation

Die Abhängigkeit der numerischen Approximation an eine analytische Differentiation (Formel (1)) von der Schrittweite h soll anhand der Funktion
$f(t) = Sin(t)$ gezeigt werden.
Hierzu vergleiche man die absoluten Werte von $\dot{f}(t_0)_{analytisch}$ mit $\dot{f}(t_0)_{numerisch}$ und berechne die relative Abweichung $r(h)$:

$$r(h) = log_{10}|\frac{\dot{f}(t_0)_{analytisch} - \dot{f}(t_0)_{numerisch}}{\dot{f}(t_0)_{analytisch}}| \tag{11}$$

an der Stelle $t_0 = 5$ für mindestens 100 verschiedene Schrittweiten h zwischen 1 und 10^{-6} für einfache Genauigkeit (in Fortran entspricht dies dem Variablentyp *real*) und h zwischen 1 und 10^{-15} für doppelte Genauigkeit (in Fortran entspricht dies dem Variablentyp *real*8*). Diese Ergebnisse sind graphisch darzustellen. Welchen Einfluss hat die Schrittweite auf die Approximation? Was passiert bei kleinen Schrittweiten?
Tipp: Tragen Sie auch die Schrittweite logarithmisch auf! Mehr Werte für kleine Schrittweiten sind daher besser als eine Gleichverteilung über den gesamten Bereich!

[4] ansonsten müsste man in den Gleichungen (10) erneut $\simeq$ anstelle von $=$ verwenden.

3.2 Numerische Integration

Bestimmen Sie analog zum vorhergehenden Beispiel die Güte der numerischen Näherung an die Stammfunktion $F(t)$ der trigonometrischen Funktion $f(t) = \frac{dF(t)}{dt} = Sin(t)$, indem Sie die Werte von $F(t)_{analytisch}$ mit $F(t)_{numerisch}$ vergleichen und berechnen die relative Abweichung $R(h)$:

$$R(h) = log_{10}|\frac{F(t)_{analytisch} - F(t)_{numerisch}}{F(t)_{analytisch}}| \tag{12}$$

an der Stelle $t = 5$ für mindestens 100 verschiedene Schrittweiten h zwischen 1 und 10^{-6} für einfache Genauigkeit und h zwischen 1 und 10^{-15} für doppelte Genauigkeit. Berechnen Sie dabei jeweils lediglich *einen* Integrationsschritt mit Schrittweite h (Formel (7))! Welcher Unterschied fällt Ihnen beim Vergleich mit der numerischen Differentiation auf?

3.3 Numerische Integration des Harmonischen Oszillators

Lösen Sie die DGL des Harmonischen Oszillators:

$$\ddot{x}(t) = -x(t)$$

mittels numerischer Integration für $t - t_0 = n \cdot \bar{h} = 100$ Zeiteinheiten (siehe Gleichung (9)), mit Anfangsbedingungen $x(t_0) = 1,\ \dot{x}(t_0) = 0$ für konstante Schrittweiten $\bar{h} = \{0.5, 0.1, 0.01, 0.001\}$ (Formeln (10))! Erzeugen Sie zu jeder Integration mit einer bestimmten Schrittweite ein Phasenraumdiagramm (x gegen $\dot{x}$ für jeden Zeitpunkt t_i) und vergleichen Sie dessen Verhalten mit der analytischen Lösung! Berechnen Sie auch die Gesamtenergie $E = T + U$ ($T =$ kinetische und $U =$ potentielle Energie) des Harmonischen Oszillators zu jedem Zeitpunkt t_i der Integrationen! Welche Unterschiede finden Sie zum analytisch vorhergesagten Wert? Wie hängen diese mit der gewählten Schrittweite zusammen?
Tipp: Spalten Sie die DGL zweiter Ordnung in ein System von DGLen erster Ordnung:

$$\dot{x}(t) = v(t)$$
$$\dot{v}(t) = -x(t)$$

Diese können Sie problemlos gleichzeitig[5] mit dem Eulerverfahren integrieren.

$$x(t_i) = x(t_{i-1}) + \bar{h} \cdot \dot{x}(t_{i-1})$$
$$v(t_i) = v(t_{i-1}) + \bar{h} \cdot \dot{v}(t_{i-1})$$

(Diese Formeln sind als numerische Wertzuweisungen zu verstehen, da sie mathematisch exakt nur für kleine Schrittweiten $\bar{h}$ mit $i \in \mathbb{N}_+$ gelten. $\rightarrow \simeq$ anstatt $=$, eh scho wissen.)
Tipp: Es empfielt sich aus Speicherplatzgründen *nicht* den gesamten Verlauf von $x(t_i)$ und $v(t_i)$ in einem Array zu speichern. Stattdessen kann man aus den unmittelbar vorhergehenden Werten $x(t_{i-1})$ und $v(t_{i-1})$ die aktuellen Werte $x(t_i)$ und $v(t_i)$ berechnen und gleich in eine Datei ausgeben. Die Werte von $x(t_i)$ und $v(t_i)$ werden im nächsten Schritt den Variablen $x(t_{i-1})$ und $v(t_{i-1})$ zugewiesen, um mit deren Hilfe erneut $x(t_i)$ und $v(t_i)$ zu ermitteln.

[5] Die neuen Orte $x(t_i)$ und Geschwindigkeiten $v(t_i)$ müssen jeweils im *selben* Integrationsschritt berechnet werden!

4 Bonus

Führen Sie - analog zur numerischen Integration - eine Fehlerabschätzung der numerischen Näherung zur ersten Ableitung durch (2 Punkte)!
Entwickeln Sie Verfahren *zweiter Ordnung* zur numerischen Differentiation und Integration. Für jedes korrekt formulierte Verfahren (inklusive Herleitung) erhalten Sie 2 Punkte, für jedes im Programmcode implementierte Verfahren zusätzliche 3 Punkte.
Tipp: Werfen Sie einen Blick in die Vorlesungsunterlagen.

5 Requirements

Ein Protokoll, kein Märchen

- das eine Einleitung zur Problemstellung enthält
- in dem alle im Text gestellten Fragen beantwortet werden
- das keinen Quellcode enthlält; das Programm wird als Textdatei **FEHLERFREI COMPILIERBAR** (MIT GFORTRAN ODER EINEM DER GEWÄHLTEN PROGRAMMIERSPRACHE ENTSPRECHENDEN GNU COMPILER!) zusammen mit Namen und Matrikelnummer an *siegfried.eggl@univie.ac.at* geschickt.
- das Ergebnisse präsentiert und analysiert.

Das Protokoll kann als PDF ebenfalls an *siegfried.eggl@univie.ac.at* versandt werden. Keine MS-Word Dokumente als mail → ausgedruckt vorlegen!

Literatur

Cash, J. R., Karp, A. H.: *A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides* ACM Transactions on Mathematical Software, Vol. 16, No. 3, *p. 201-222* (1990)

Vesely, F.: *Computational Physics - An Introduction* Springer US, *p.105 et sqq.* (2001)

Wikipedia, *Fundamental theorem of calculus*, `http://en.wikipedia.org/wiki/Fundamental_theorem_of_calculus` (2009)