Computergestütztes wissenschaftliches Rechnen
der Fakultät für Physik,
Universität Göttingen

CWR2 Abschlussarbeit SoSe 2014
Projekt 5 - Chemische Kinetik

Student:	Michael Lohmann
Matrikelnr.:	2135 2626
E-Mail:	m.lohmann@stud.uni-goettingen.de
Dozent:	Dr. Salvatore R. Manmana
	Dr. Ulrich Welling
Betreuer:	Burkhard Blobel
Abgabedatum:	18.8.2014

Note:

Inhaltsverzeichnis

1 Aufgaben

Die Bewegung von Teilchen zu bestimmen ist eine wichtige Voraussetzung um Prognosen zu erstellen, wie sich ein System verhält. Insbesondere ist es interessant (nicht nur für Chemiker) die Entwicklung der Dichte zweier verschiedener Stoffsorten, welche vermischt sind, zu beobachten, da diese in großem Maße für deren Reaktionsfähigkeit entscheidend sind. Die Gleichungen, welche dieses System beschreiben, lauten

$$\frac{du(t)}{dt} = a - u(t) + u^2(t) \cdot v(t) = f_u(u,t) \tag{1}$$

$$\frac{dv(t)}{dt} = b - u^2(t) \cdot v(t) = f_v(v,t) \tag{2}$$

wobei $u(t)$ und $v(t)$ die Dichten der beiden Molekülsorten sind und a und b zwei Konstanten, welche größer 0 sind. Außerdem kann die Dichte eines Stoffes natürlich nicht negativ werden, so dass $u(t)$ und $v(t)$ ebenfalls immer positiv sein müssen.

Ein selbstgeschriebenes Programm soll die Startwerte und Parameter einlesen und daraus mithilfe des Euler- sowie des Runge-Kutta-Algorithmus die zeitliche Entwicklung berechnen.
Für ein fest gewähltes $b < 1$ sollen für mehrere a Diagramme erstellt werden, die $v(t)$ und $u(t)$ gegen t auftragen, sowie eins, welches $u(t)$ gegen $v(t)$ darstellt.
Des weiteren ist es interessant, für welche Parameter a und b es für große Zeiten zu periodischen Lösungen kommt.

2 Runge-Kutta-Algorithmus

Der Runge-Kutta-Algorithmus ist ein Ansatz, Differentialgleichungen numerisch zu lösen. Er ist eine einfache und dabei relativ robuste Möglichkeit, Näherungen zu bekommen. Der wohl am häufigsten benutzte ist dabei derjenige 4. Ordnung. In der diskretisierten Form berechnet sich das folgende Glied aus dem vorherigen nach [1, S.130] durch

$$\boxed{y_{i+1} = y_i + (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4)/6}$$

mit

$$\begin{aligned} k_1 &= \Delta t \cdot f(y_i,\, t_i) \\ k_2 &= \Delta t \cdot f(y_i + k_1/2,\, t_i + \Delta t/2) \\ k_3 &= \Delta t \cdot f(y_i + k_2/2,\, t_i + \Delta t/2) \\ k_4 &= \Delta t \cdot f(y_i + k_3,\, t_i + \Delta t) \end{aligned}$$

wobei $\dot{y} = f(y,t)$.
Für kompliziertere Differentialgleichungen ist eventuell eine höhere Ordnung nötig, oder

aber eine komplett andere Herangehensweise, die auch mit komplexeren DGLs zurechtkommt. Für viele Probleme reicht aber der Runge-Kutta-Algorithmus 4. Ordnung aus. Der Euler-Algorithmus berechnet den nächsten Schritt nur mit $y_{i+1} = y_i + k_1$ und ist dadurch in der Regel ungenauer (gleiche Ergebnisse lassen sich nur bei Funktionen mit konstanter Ableitung erwarten).

In unserem Fall haben wir zwei gekoppelte Differentialgleichungen erster Ordnung. Bei der Berechnung der Koeffizienten muss hierbei darauf geachtet werden, dass sich beide Parameter mit fortlaufender Zeit ändern. In unserem Fall berechnen sie sich so:

$$
\begin{aligned}
k_{1_u} &= \Delta t \cdot f_u(u_i,\, v_i) \\
k_{1_v} &= \Delta t \cdot f_u(v_i,\, u_i) \\
k_{2_u} &= \Delta t \cdot f_u(u_i + k_{1_u}/2,\, v_i + k_{1_v}/2) \\
k_{2_v} &= \Delta t \cdot f_v(v_i + k_{1_v}/2,\, u_i + k_{1_u}/2) \\
k_{3_u} &= \Delta t \cdot f_u(u_i + k_{2_u}/2,\, v_i + k_{2_v}/2) \\
k_{3_v} &= \Delta t \cdot f_v(v_i + k_{2_v}/2,\, u_i + k_{2_u}/2) \\
k_{4_u} &= \Delta t \cdot f_u(u_i + k_{3_u},\, v_i + k_{3_v}) \\
k_{4_v} &= \Delta t \cdot f_v(v_i + k_{3_v},\, u_i + k_{3_u})
\end{aligned}
$$

3 Programmstruktur

Das Programm beginnt mit der Definition der beiden Funktionen f_u und f_v, welche die jeweiligen Werte aus den Daten von u bzw. v berechnen. In der *main* werden zunächst die Variablen deklariert und soweit möglich initialisiert. Folgend werden die restlichen per Benutzereingabe eingelesen, so man nicht einen automatischen Durchlauf mit variablen Parametern wünscht und dies auskommentiert. Dabei wird auch überprüft, ob sie der Voraussetzung entsprechend > 0 sind. Sollte dies nicht der Fall sein, so werden sie so lange erneut abgefragt, bis sie > 0 sind. Sind alle 4 Werte korrekt eingegeben, so werden sie in die erste Stelle der Arrays eingetragen. Dies sind 4 Stück: je eins für u und eins für v zu den beiden Algorithmen.
Eine zunächst auskommentierte Schleife beinhaltet die Möglichkeit, für verschiedene b und a die Berechnung automatisiert durchzuführen.
Im folgenden wird mit einer Hilfskonstruktion aus Stringstreams und Strings der Dateiname für die spätere Ausgabedatei erstellt. Diese wird nun zum Schreiben geöffnet und in ihre ersten beiden Zeilen werden die Beschriftungen der Spalten und die Startparameter eingetragen.
Dann beginnt die Berechnung. Eine Schleife geht jeden Zeitschritt durch bis die Endzeit erreicht ist. Zunächst wird in die erste Spalte der aktuellen Schritt geschrieben. Aus diesem wird nun der Wert für den darauf folgenden berechnet. Dieser wird zunächst mit dem Euler-Verfahren und dann nach Runge-Kutta ausgewertet und die aktuellen

Ergebnisse werden in die Ausgabedatei geschrieben.
Da u und v des nächsten Schritts jedoch voneinander abhängen, kann der Runge-Kutta-Algorithmus nicht zunächst die eine und dann die andere Berechnung machen. Das Problem wird dadurch gelöst, dass die Koeffizienten k_{1_u} bis k_{4_u} immer abwechselnd mit k_{1_v} bis k_{4_v} berechnet werden und die Zwischenergebnisse mit in die nächste Berechnungen mit einfließen. Diese werden wie auch beim Euler-Verfahren jeweils mithilfe der als erstes im Programm definierten Funktionen f_u und f_v berechnet. Die dafür nötigen Werte werden ihnen übergeben und der Rückgabewert wird mit Δt multipliziert um die Koeffizienten zu erhalten. Sind alle berechnet, so wird aus diesen der nächste Schritt berechnet. Wurden alle gefragten Zeitschritte berechnet, so wird die Datei geschlossen.
Anschließend erstellt das Programm eine Rahmendatei für Gnuplot und führt diese aus. Nachdem der Plot für 10 Sekunden angezeigt wurde, (unter Windows könnte das automatische Anzeigen nicht funktionieren, da meines Wissens nach Gnuplot nicht über die Konsole gesteuert wird) wird das Programm beendet.

4 Auswertung

Wie man in den Abb. 1(a) bis 2(c) sieht, nimmt mit größer werdendem Parameter a nicht nur der Grenzwert von u zu (von v ab), sondern die Schwingung von u und v wird im Verlauf der Zeit deutlich schneller gedämpft. In den Abbildungen kann man auch gut erkennen, dass die Werte für kleine a durch ihren Verlauf herausstechen. Sie konvergieren nicht, sondern schwingen immer und so gehe ich davon aus, dass für kleine a der Algorithmus keine sinnvollen Ergebnisse bringt. Für $b = 0.8$ liegt diese Grenze bei ca. $a = 0.09$ wohingegen sie für $b = 0.4$ schon bei $a = 0.2$ liegt Dies habe ich durch betrachten der Graphen für unterschiedlichste Parameter ermittelt. Der Runge-Kutta-Algorithmus scheint jedoch für große b stabiler zu sein und weniger fehleranfällig bezüglich dieses Problems, da ein großes b dämpfend wirkt.

5 Stabilität der Algorithmen

Es ist keinesfalls belibig, welchen Algorithmus man verwendet. Dies kann man insbesondere an der Abbildung 3 erkennen, welche in Teil a) für Schrittweiten von $\Delta t = 0.1$ die Ergebnisse des Runge-Kutta-Algorithmus zu den Eingabewerten $a = 0.01, b = 0.99, u_0 = v_0 = 1$ zeigt. Hierbei ist eine deutliche Konvergenz zu einem Grenzwert von 1 erkennen. Guckt man sich das zweite Bild an, welches zusätzlich noch für die selben Startwerte den Euler-Algorithmus auswertet, so erkennt man, dass komplett gegenteilige Aussagen entstehen. Der grün dargestellte Euler-Algorithmus divergiert bis er konstante Schwingungen bei einer Amplitude von 1.2 ausführt.
Die Startwerte für diese Bilder habe ich durch wiederholtes Ausprobieren ermittelt, bis ein möglichst eindeutiges Bild entstand.

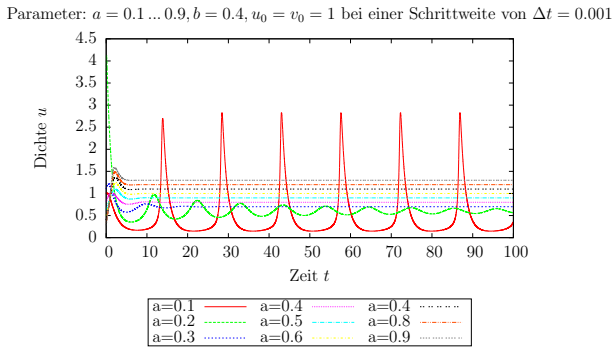
6 Mögliche Verbesserungen

Als zusätzliche Verbesserung des Runge-Kutta-Algorithmus 4. Ordnung mit konstanten Zeitschritten wäre es möglich, die Zeitschritte je nach Zeitpunkt unterschiedlich zu wählen. Es ist z.B. wichtig, wenn sich die Funktion schnell ändert, möglichst kleine Schritte zu machen. Ändert sich jedoch wenig, so ist eine kleine Schrittweite nur rechenintensiv und trägt nur unwesentlich zu einer genaueren Berechnung bei.

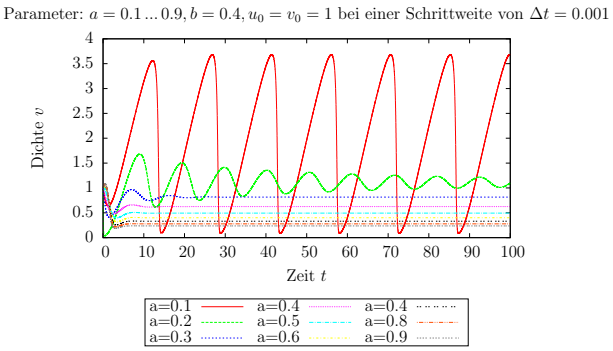
Anhang

Literatur

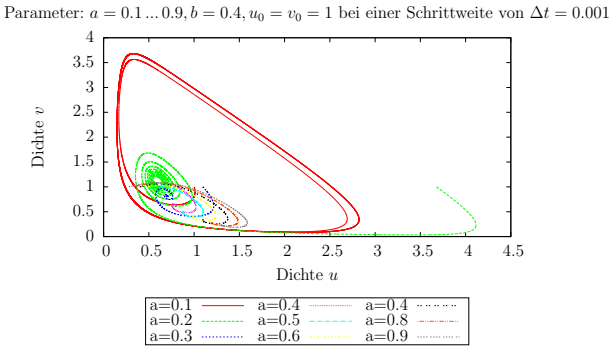
[1] J. Pitt-Francis and J. Whiteley (2012): *Guide To Scientific Computing in C++*, 1. Auflage, Springer London



(a) $u : t$

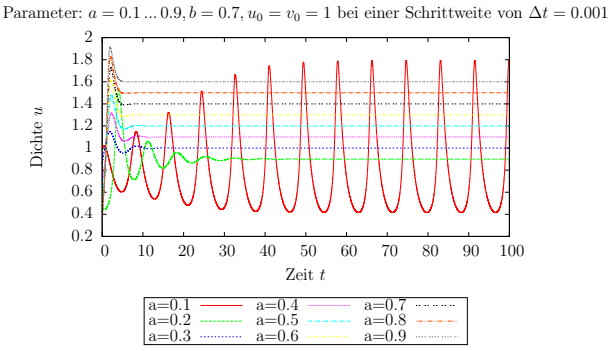


(b) $v : t$

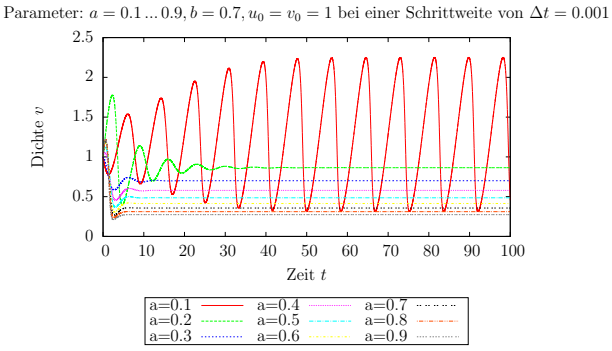


(c) $u : v$

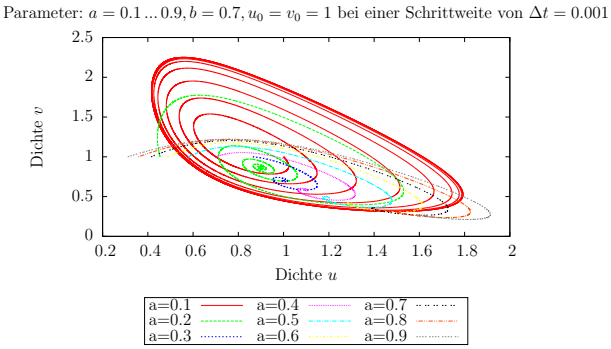
Abbildung 1: u und v gegen t und u gegen v aufgetragen für $b = 0.4$



(a) $u : t$

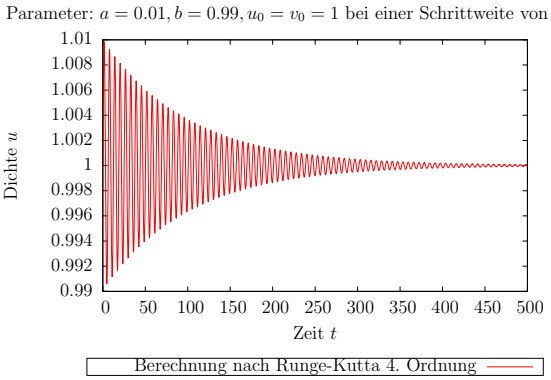


(b) $v : t$

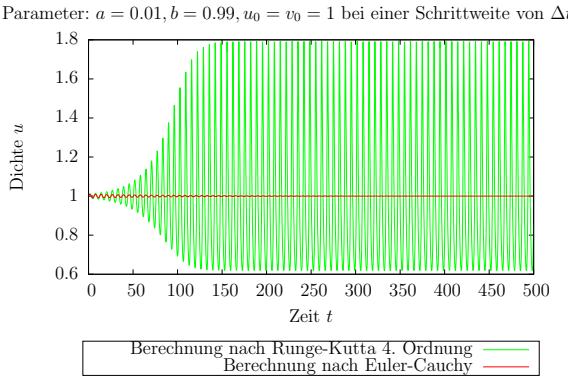


(c) $u : v$

Abbildung 2: u und v gegen t und u gegen v aufgetragen für $b = 0.7$



(a) Runge-Kutta-Algorithmus 4. Ordnung



(b) Vergleich Runge-Kutta-Algorithmus mit Euler bei identischen Randbedingungen

Abbildung 3: Unterschiedliche Algorithmen im Vergleich