Programmieren im Zeitalter des Internet

„Es gibt nichts Gutes, außer man tut es“ (Erich Kästner)

Der vorliegende Gesamtkurs „Programmierung naturanaloger Verfahren“ ist eine Online Version unseres gleichnamigen Buches in Form eines Kurses, das 2010 im Verlag Vieweg + Teubner erschienen ist. Dies Buch erhalten Sie als Teilnehmer an einem Kurs unentgeltlich, d.h. ohne zusätzliche Gebühren. Der Gesamtkurs ist in vier Teilkurse gegliedert, die unten kurz benannt werden. Sie können sowohl den Gesamtkurs belegen, als auch einen oder mehrere der Teilkurse; die Kursgebühren und die Leistungsanforderungen sind entsprechend gestaffelt.

Die Möglichkeit, nur einzelne Teilkurse zu belegen, haben wir ausdrücklich für Teilnehmer vorgesehen, die sich nur für eines oder zwei der von uns dargestellten Themen interessieren. Universitär zertifiziert werden alle Teilkurse ebenso wie der Gesamtkurs mit den entsprechenden speziellen Themenangaben.

Warum aber ein Programmierkurs im Zeitalter des Internet?

Zu Beginn eine kleine und leider wahre Geschichte: In einer mündlichen Prüfung stellte ein Student einen von ihm selbst programmierten Genetischen Algorithmus vor. Als er darauf hinwies, dass er das sog. Roulette-Wheel Verfahren als Heiratsschema verwendet hatte, fragten wir ihn, warum er dies Verfahren gewählt hatte und wie es definiert ist (die Fachausdrücke werden im einschlägigen Teilkurs über evolutionäre Algorithmen kurz erläutert). Beide Fragen konnte der Prüfling nicht beantworten; bei Nachhaken unsererseits stellte sich heraus, dass der Student nicht nur dies Verfahren, sondern einen Großteil seiner Subprogramme einfach aus dem Internet übernommen hatte, ohne sich weitere Gedanken dazu zu machen. Anscheinend hatte er nicht erwartet, dass er sein eigenes Programm auch noch verstehen sollte. Zur relativen Ehrenrettung dieses anonym bleibenden Studenten sei hinzugefügt, dass er bei weitem nicht der Einzige war, bei dem wir derartige Prüfungserlebnisse hatten.

Nun ist es eine bekannte Tatsache, dass häufig Studierende eine arbeitssparende Subspezies der Gattung Homo Sapiens darstellen; einer der Autoren bekennt reuig, dass er selbst im Verlauf seines Mathematikstudiums regelmäßig Lösungen von Beweisaufgaben von großzügigen Kommilitonen schlicht abgeschrieben hat (diese freilich auch von ihm). Als Wissenschaftler und Lehrende wissen wir allerdings auch, wie fatal ein derartiges Verhalten ist, wenn es darum geht, eigene Programme zu konstruieren und diese in den logischen Grundzügen nicht zu verstehen. Dem praktischen Desaster sind damit Tür und Tor geöffnet.

Insbesondere ist es dann auch nicht möglich, die Ursache für Fehler des Programms zu erkennen, geschweige denn die Fehler zu beseitigen.

Es ist ein angenehmer Umstand, dass das Internet nicht nur alle möglichen Informationen zur Verfügung stellt, sondern dass auch in zunehmendem Maße ganze Programmteile in Form von Klassen, Prozeduren und Bibliotheken unmittelbar aus dem Internet übernommen werden können. Sofern es sich dabei um Routineangelegenheiten handelt, ist dagegen auch gar nichts zu sagen. Das gilt auch und insbesondere für die Techniken, um deren Programmierung es in diesem Buch geht. Wenn jedoch eine Übernahme vorgefertigter Lösungen nicht einfach nur lästige aber durchaus verstandene Routine ersparen soll, sondern das eigene Verständnis und das eigene Denken ersetzt, dann wird es in jeder Hinsicht kritisch.

Aus Jahrhunderten didaktischer Praxis und allen möglichen Lehr- und Lernerfahrungen ist hinlänglich bekannt, dass das eigene Verständnis am besten dadurch realisiert wird, indem man nicht einfach rezeptiv kluge Worte und Orientierungen geduldig übernimmt, sondern selbst praktische Erfahrungen sammelt, also etwas tut. Wir haben deswegen den Gesamtkurs folgendermaßen konzipiert:

In jedem Einzelkurs erhalten Sie in Form von Stichworten kurze Erläuterungen, worum es jeweils geht. Was also ist ein Zellularautomat, was sind neuronale Netze, was sind evolutionäre Algorithmen usf. Es muss jedoch noch einmal betont werden, dass es sich hier nur um Stichworte handelt. Für genauere Beschreibungen dieser künstlichen Systeme und deren theoretische Analysen verweisen wir ausdrücklich auf unser Buch „Modellierung komplexer Prozesse durch naturanaloge Verfahren“, Vieweg + Teubner 2009 (nicht identisch mit dem Buch, das Sie von uns erhalten). Anschließend wird gezeigt, wie zuerst die logischen Grundbausteine der jeweiligen Techniken programmiert werden können, also z.B. eine Zelle und ein Zellengitter bei Zellularautomaten oder ein künstliches Neuron für ein neuronales Netz. Der nächste Schritt ist dann die Kombination der Grundbausteine zu einfachen Programmteilen wie z.B. die Anwendung der sog. genetischen Operatoren auf künstliche Chromosomen, dargestellt als Vektoren, bei evolutionären Algorithmen. Am Ende folgen dann in Form von Beispielen ganze Programme, die noch einmal die Programmierlogik für die jeweiligen Techniken darstellen. Um Ihnen die erwähnte eigene Tätigkeit zu ermöglichen, haben wir zuweilen verschiedene Übungen angegeben, an denen Sie erproben können, inwiefern Sie unsere Hinweise in eigene Programme umsetzen können. Aus leidvollen Erfahrungen bei der Realisierung eigener Programme wissen wir, was alles dabei schief gehen kann bzw. gemäß dem Gesetz von Murphy ziemlich sicher schief gehen wird. Die Bearbeitung dieser Übungen ist allerdings nicht obligatorisch.

Neben den oben erwähnten eher melancholisch stimmenden Erfahrungen haben wir natürlich auch jede Menge von engagierten Studierenden erlebt, die sehr gerne selbst Programme realisieren wollten. Nicht zufällig haben wir dabei nicht nur von Programmieranfängern sondern auch von erfahrenen Programmierern gehört, dass die Programmierung der von uns thematisierten Techniken alles andere als selbstverständlich ist. Die Programmierungsprobleme bei naturanalogen Verfahren, die häufig auch terminologisch sehr unglücklich als „Soft Computing" bezeichnet werden, weichen nun einmal von den Standardaufgaben in der Informatik zuweilen sehr deutlich ab. Das ist sogar der zentrale Grund, warum wir uns entschlossen haben, das Buch zu schreiben, das Sie erhalten, und außerdem diesen Onlinekurs anzubieten: Immer wieder wurden wir von Studierenden sowohl an unserer Universität als auch in Onlinekursen gefragt, ob wir ihnen nicht einschlägige Lehrbücher für das Programmieren dieser Techniken nennen könnten. Das konnten wir nicht, da es keine gab, bis auf einige wenige veraltete Ausnahmen. Auch ein Rezensent unseres Buchs von 2009 bei Amazon bemerkte bedauernd, dass es in unserem von ihm sonst sehr gelobten Buch keine Programmbeispiele gibt. Also wandten wir das Zitat von Kästner kurzerhand auf uns selbst an; das Ergebnis liegt vor Ihnen. Wir hoffen, auch den Amazon-Rezensenten damit nachträglich zufrieden stellen zu können; vielleicht wird er motiviert, an diesem Kurs teilzunehmen.

Aus diesem Grund geben wir Ihnen in dem Buch und in dem Kurs auch keine weiterführenden Literaturhinweise, was zugegebenermaßen etwas ungewöhnlich ist. Zum einen, wie bemerkt, gibt es keine lohnenswerte Literatur für Programmiertechniken auf unserem Gebiet; zum anderen haben wir in unserem ersten Buch von 2009 zahlreiche allgemeine Literaturhinweise zu den einzelnen Gebieten angegeben. Wir verweisen nur dort auf Literatur, wo wir unmittelbar etwas von anderen Autoren übernommen haben. Allerdings, übernehmen Sie dies Verfahren bitte nicht unbedingt für Ihre eigenen Publikationen oder studentischen Arbeiten.

Eine weitere kleine Geschichte aus unserer langjährigen Lehrpraxis soll Ihnen nicht vorenthalten werden: In einer anderen Prüfung stellte ein Student ein von ihm programmiertes neuronales Netz vor und erklärte, dass er eine neue Version der sog. Backpropagation-Regel verwendet hatte (s. den Teilkurs über neuronale Netze). Es überrascht Sie jetzt wohl nicht mehr, dass besagter Student als fleißiger Benutzer des Internet nicht nur nicht sagen konnte, was denn der Vorteil seiner Version gegenüber der Standardregel ist, sondern dass er auch das mathematische Verfahren dieser Standardlernregel in keiner Weise verstanden hatte. Nun muss man diesem Studenten konzedieren, dass diese Regel in der Tat nicht ganz einfach zu verstehen ist und vor allem dass in den üblichen Lehrbüchern die

mathematischen Grundlagen derartiger Algorithmen wenn überhaupt nur kursorisch dargestellt und praktisch nie erklärt werden.[1]

Wir haben uns deswegen entschlossen, in mehren Fällen genauer auf mathematische Grundlegen und spezielle Berechnungsverfahren einzugehen, insbesondere auch auf besagte Lernregel. Uns ist natürlich bewusst, dass dies manche Leser abschrecken kann, da Mathematik auch für Informatiker und andere an Programmiertechniken interessierte Studierende und Praktiker häufig leider abschreckend wirkt. Der große britische Mathematiker und Physiker Roger Penrose illustrierte dies einmal sehr treffend mit der Warnung seines Verlegers, dass jede Formel in einem Buch die Anzahl der potentiellen Leser jeweils um die Hälfte reduziert. Wegen der genannten Erfahrungen gehen wir trotzdem dies Risiko ein und verweisen auf den entsprechenden Hinweis von Penrose an seine Leser: Wenn Sie an den mathematischen Grundlagen nicht interessiert sind sondern vor allem an den Programmiertechniken, dann können Sie die mathematischen Darstellungen ohne jedes Problem überspringen und sich ganz dem Programmieren widmen. Vielleicht haben Sie nach der Lektüre des gesamten Buchs dann doch Lust, sich den mehr mathematisch orientierten Textteilen zuzuwenden.

Leider ist hier eine kleine Warnung am Platze: aufgrund der zahlreichen verschiedenen Betriebssysteme und der zahlreichen Programmiersprachen kann es durchaus geschehen, dass Sie unsere Berechnungsergebnisse nicht immer ganz nachvollziehen können. Falls dies geschieht, liegt die Quelle der Abweichungen gewöhnlich bei den unterschiedlichen Rundungsverfahren, mit denen jeweils gearbeitet wird. Sofern Sie also Abweichungen erzielen, müssen Sie durchaus keine Fehler gemacht haben. Teilen Sie uns dies dann bitte mit.

Strukturell folgt dieser Kurs dem Ihnen vorliegenden Buch, so dass Sie, falls Sie wollen, die beiden Texte parallel lesen können. Drei wichtige Hinweise dürfen allerdings nicht fehlen:

Wir erleben seit einiger Zeit eine Inflation von neuen Programmiersprachen. Auch wenn viele von ihnen eigentlich „nur“ Varianten bzw. Weiterentwicklungen bestimmter Standardsprachen wie z.B. C-# oder JAVA sind, unterscheiden sie sich leider dann doch immer wieder in verschiedenen Aspekten. In einem Buch und erst recht einem Kurs über Programmiertechniken ist es deswegen völlig unmöglich, die Beispiele für Codes in allen wichtigen Sprachen zu geben. Das würde Buch und Kurs rasch unlesbar machen und vermutlich auch jeden akzeptablen Umfang sprengen. Aus diesen und anderen Gründen haben wir uns für ein anderes Vorgehen entschlossen: Fast alle Beispiele sind in Form von

[1] Für den Fall der Backpropagation-Regel müssen wir selbstkritisch anmerken, dass auch wir diese in unserem erwähnten Buch von 2009 nicht exakt erklärt bzw. mathematisch abgeleitet haben.

kommentierten Pseudocodes in einer an FORTRAN orientierten Darstellung gegeben. Wie der Name schon sagt, nämlich Formula Translation, handelt es sich um eine Sprache, die sich (noch) sehr stark an den mathematischen Darstellungsweisen orientiert; diese sind nun einmal nach wie vor *der* universale Formalismus.[2] Wir glauben, dass unsere Beispiele auch für Leser relativ leicht zu verstehen sind, die sich in Fortran 95 selbst nicht auskennen. Um diesen Lesern jedoch das Verständnis zu erleichtern, haben wir im Anhang A des Buches Hinweise gegeben, wie die einzelnen von uns verwendeten Zeichen zu verstehen sind. Bevor Sie sich also mit den einzelnen Programmbeispielen vertraut machen, sollten Sie sich diesen Anhang zu Gemüte führen.

Programmiersprachen kommen und gehen, mathematische Symbolismen und Formalismen jedoch bleiben. Deswegen, so hoffen wir, sind unsere Darstellungen in FORTRAN auch dann noch verwendbar, wenn die gegenwärtig aktuellen Sprachen schon wieder vergessen sind. Für diejenigen Leser, die mit mehr objektorientierten Sprachen vertraut sind, haben wir immer wieder einschlägige Hinweise gegeben.

Komplementär zu unserer Bevorzugung „zeitloser" Darstellungen kann es natürlich auch recht nützlich sein, einige unserer Programmbeispiele in aktuellen Sprachen wie einer JAVA Version oder C-# zu studieren. Verschiedene unserer Studenten haben angeboten, derartige Beispiele zu programmieren und auf unsere Homepage zu stellen. Klicken Sie also, wenn Sie daran interessiert sind, dort einmal an und erfreuen sich an dem Engagement unserer Studenten: www.cobasc.de.

Ein zweites Problem ist das der Anlage und Durchführung von Experimenten. Wir empfehlen nicht nur in diesem Buch immer wieder, die eigenen Programme durch Experimente zu überprüfen bzw., wie man so schön sagt, zu validieren. Wenn man jedoch nicht aus einer Experimentalwissenschaft kommt, ist nicht unbedingt klar, wie man Experimente anlegt und auswertet. Auch das haben uns unsere Erfahrungen mit vielen Studenten gelehrt. Im Anhang B haben wir deswegen einige grundsätzliche Hinweise zu diesem Thema gegeben. Wenn Sie also Experimente mit einem Ihrer eigenen Programme durchführen wollen und von Ihrem intellektuellen Hintergrund her mit experimentellen Verfahren nicht vertraut sind, sollten Sie sich diesen Anhang ebenfalls zu Herzen nehmen.

Schließlich möchten wir darauf hinweisen, dass im Kurs zwar Einiges nicht vorkommt, das Bestandteil des Buches ist. Dafür finden sich im Kurs verschiedene Beispiele, in denen einige von uns entwickelte Programme in ihren Endversionen „dynamisch" dargestellt worden sind, nämlich in Form von Videos. Diese durch

[2] Deswegen wird FORTRAN vor allem von Naturwissenschaftlern und Mathematikern benutzt.

die Weiterentwicklung der PDF-Formate ermöglichte Darstellungsform findet sich naturgemäß in unserem Buch nicht und so ist der Kurs auch reicher als das Buch.

Ein rechtlicher Hinweis darf leider nicht fehlen:

Wir weisen darauf hin, dass die Fehlerfreiheit der Code-Beispiele nicht vollständig garantiert werden kann und dass wir keine Haftung für eventuelle Schäden übernehmen, die bei der Verwendung der Code-Beispiele entstehen.

Hinsichtlich der Leistungsanforderungen für ein erfolgreiches Absolvieren des Gesamtkurses oder einzelner Teilkurse gilt Folgendes:

Für jeden Teilkurs muss eine Programmieraufgabe gelöst werden. Falls Sie also den Gesamtkurs wählen, müssen vier Aufgaben abgegeben werden. Bei den Teilkursen, in denen zwei verschiedene Bereiche behandelt werden, können Sie wählen, welche der beiden von uns gestellten Aufgaben gelöst werden soll. Falls Sie eine eigene Thematik bei den Aufgaben behandeln wollen, können Sie dies tun, aber nur nach Absprache mit uns.

Bei der Abgabe der Aufgabe(n) müssen Sie auf folgende Bedingungen achten:

Abgegeben werden müssen a) der Quellcode mit kurzen Kommentaren, b) eine Darstellung der Aufgabe und Ihrem Programm in einen eigenen Dokument (also nicht als Teil des Quellcodes) und c) eine Darstellung der von Ihnen durchgeführten Experimente mit Ihrem Programm sowie die wichtigsten Ergebnisse ebenfalls in einem eigenständigen Text. Falls Sie aufgrund der Absprache mit uns ein eigenes Thema bearbeitet haben, muss dieses im Dokument b) natürlich inhaltlich dargestellt und kurz begründet werden –nämlich warum Sie das Thema gewählt haben. Fragen zu diesen Anforderungen können Sie natürlich jederzeit mit uns besprechen.

Abschließend wünschen wir Ihnen möglichst viel Vergnügen daran, selbst zu programmieren und Ihre Programme zu erproben.