
Physik mit Python

Oliver Natt

Physik mit Python

Simulationen, Visualisierungen
und Animationen von Anfang an

2. Auflage

Oliver Natt
Technische Hochschule Nürnberg
Georg Simon Ohm
Nürnberg, Deutschland

ISBN 978-3-662-66453-7 ISBN 978-3-662-66454-4 (eBook)
<https://doi.org/10.1007/978-3-662-66454-4>

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© Springer-Verlag GmbH Deutschland, ein Teil von Springer Nature 2020, 2022

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von allgemein beschreibenden Bezeichnungen, Marken, Unternehmensnamen etc. in diesem Werk bedeutet nicht, dass diese frei durch jedermann benutzt werden dürfen. Die Berechtigung zur Benutzung unterliegt, auch ohne gesonderten Hinweis hierzu, den Regeln des Markenrechts. Die Rechte des jeweiligen Zeicheninhabers sind zu beachten.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Planung/Lektorat: Gabriele Ruckelshausen

Springer Spektrum ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.

Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Vorwort

Es ist weitgehend unbestritten, dass der Computer zu einem nahezu unverzichtbaren Werkzeug in den Ingenieur- und Naturwissenschaften geworden ist, und auch in den allgemeinbildenden Schulen wird mit größer werdendem Druck eine Digitalisierung des Unterrichts gefordert. An Universitäten und Hochschulen werden schon seit längerer Zeit viele Lehrveranstaltungen über Physik durch Computergrafiken, -animationen und -simulationen ergänzt. Dabei werden die oftmals schönen Animationen zwar von den Studierenden mit Begeisterung aufgenommen, der Erkenntnisgewinn durch das bloße Betrachten der Animationen ist dagegen oft gering.

Im Gegensatz dazu berichten Lehrende immer wieder, dass sie selbst beim Erstellen der Animationen und Grafiken erstaunlich viel hinzulernen. Es klingt daher verlockend, entsprechende Programmieraufgaben zu stellen, damit nicht dem Lehrenden, sondern den Studierenden dieser Lernerfolg zuteilwird. Leider scheitert dies oft an den Programmierkenntnissen. Die Tatsache, dass unser Alltag in hohem Maße von digitalen Geräten durchdrungen ist, darf nicht darüber hinwegtäuschen, dass die wenigsten Studierenden zu Beginn ihres Studiums mit irgendeiner Programmiersprache wirklich so vertraut sind, dass sie diese auf Anhieb dazu verwenden können, um ein gegebenes Problem zu lösen. Es stellt sich somit die Frage nach einem geeigneten didaktischen Ansatz, um in die Benutzung des Computers zur Lösung von naturwissenschaftlich-technischen Fragestellungen einzuführen.

Ein sicherlich extremer Standpunkt, der zum Teil in den Ingenieurwissenschaften vertreten wird, geht davon aus, dass es völlig ausreichend sei, wenn man die entsprechenden Simulationsprogramme bedienen kann. Es zeigt sich allerdings immer wieder, dass die meisten Menschen mit professionellen Simulationsprogrammen völlig überfordert sind, wenn sie nicht wenigstens eine grobe Idee davon haben, wie die Programme eigentlich funktionieren.

Der andere Standpunkt, der in vielen Büchern über Computerphysik oder Computational Physics vertreten wird, besteht darin, dass man zunächst einmal jede einzelne numerische Methode von der Pike auf lernen muss, bevor man diese sinnvoll anwenden kann. Es ist unbestritten wichtig, sich mit den numerischen Methoden tiefgehend auseinanderzusetzen, wenn man ernsthaft Computerphysik betreiben möchte. Für viele Studierende ist dies aber kein geeigneter Einstieg, da man auf diese Weise erst relativ spät dazu in der Lage ist, komplexere physikalische Probleme zu bearbeiten.

Vergleichen wir die Situation einmal mit einem völlig anderen Lehrgebiet: In der Schulmathematik wird das Rechnen mit reellen Zahlen ganz intuitiv gelehrt. Es würde sicherlich niemand auf die Idee kommen, den ersten Kontakt mit den reellen Zahlen über eine axiomatische Einführung herzustellen. Genauso sollte man beim wissenschaftlichen Rechnen zunächst einmal Problemlösungsstrategien verinnerlichen. Man benötigt einen intuitiven Zugang dafür, wie man überhaupt ein physikalisches Problem in ein Computerprogramm übersetzen kann, sowie einige Erfahrungen, wie man die Ergebnisse der eigenen Computerprogramme auf Plausibilität überprüfen kann. Man muss wissen, wie man die Ergebnisse der eigenen Programme grafisch ansprechend aufbereitet, und nicht zuletzt muss man eine geeignete Programmiersprache beherr-

schen. Erst danach kann man (und dann sollte man auch) tiefer einsteigen und sich intensiver mit der zugrunde liegenden numerischen Mathematik beschäftigen.

An der TH Nürnberg erhalten die Studierenden des Studiengangs »Angewandte Mathematik und Physik« bereits in den ersten beiden Semestern eine grundlegende Einführung in Programmiertechniken. Im dritten Semester wird ein Simulationsseminar angeboten, in dem die Studierenden während eines Semesters – meistens in einer Zweiergruppe – an einem Simulationsthema arbeiten. Die dabei von den Studierenden erzielten Ergebnisse begeistern immer wieder aufs Neue, und auch die Studierenden geben sehr viele positive Rückmeldungen zu dieser Lehrveranstaltung, die oft noch am Ende des Bachelorstudiums als ein »Highlight des Studiengangs« bezeichnet wird.

Dieses Buch soll mehr Studierende für das Thema Computerphysik begeistern, indem die oben beschriebene inhaltliche und didaktische Herangehensweise weiterverfolgt wird. Anhand typischer Fragestellungen aus der klassischen Mechanik wird ein praxisorientierter Einstieg in das Thema Computersimulationen gegeben.

An dieser Stelle möchte ich mich bei Frau Anja Dochnal und Frau Anja Groth für die gute Betreuung vonseiten des Springer-Verlages sowie bei Frau Margit Maly für die vielen hilfreichen Diskussionen bedanken. Ein besonderer Dank gebührt darüber hinaus den Kolleginnen und Kollegen der Fakultät »Angewandte Mathematik, Physik und Allgemeinwissenschaften« der TH Nürnberg, die mir durch Entlastung bei einigen Lehrveranstaltungen den Freiraum zum Schreiben dieses Buches gewährt haben. Nicht zuletzt möchte ich mich bei meinen Studenten, Frau Anja Mödl und Herrn Andreas Nachtmann, sowie bei Fabian Steinmeyer für das Durcharbeiten des Manuskripts und viele hilfreiche Vorschläge bedanken.

Gegenüber der ersten Auflage dieses Buchs wurden einige Fehler beseitigt, und an zahlreichen Stellen wurden kleinere Ergänzungen und Verbesserungen vorgenommen. Bei den vielen Leserinnen und Lesern der ersten Auflage bedanke ich mich ganz herzlich für die entsprechenden Hinweise. Darüber hinaus habe ich in dieser Auflage die Programme nicht nur an den aktuellen Stand der Entwicklung angepasst, sondern auch etwas mehr Wert auf formale Aspekte des Programmierens gelegt: Missverständliche Variablennamen in den Programmen wurden geändert und alle Funktionen sind jetzt durchgehend mit Docstrings versehen. Um das Buch für den Einstieg in das Programmieren noch attraktiver zu machen, wurde das einführende Kapitel über Python in zwei Kapitel aufgeteilt, sodass schon direkt nach der Einführung in die Programmiersprache einige Übungsaufgaben gestellt werden können, bevor die Bibliotheken NumPy und Matplotlib besprochen werden. Ein zusätzliches Kapitel am Ende des Buches bietet darüber hinaus einen Einblick, wie man objektorientierte Programmiermethoden für die Simulation physikalischer Probleme einsetzen kann.

Inhaltsverzeichnis

Vorwort	v
1 Einleitung	1
1.1 An wen richtet sich dieses Buch?	1
1.2 Was ist eine Simulation?	2
1.3 Die Wahl der Programmiersprache	3
1.4 Aufbau des Buches	4
1.5 Nomenklatur	5
Literatur	6
2 Einführung in Python	7
2.1 Installation einer Python-Distribution	8
2.2 Installation von Python unter Linux	9
2.3 Installation eines Texteditors	9
2.4 Installation einer Entwicklungsumgebung	10
2.5 Starten einer interaktiven Python-Sitzung	10
2.6 Python als Taschenrechner	11
2.7 Importieren von Modulen	15
2.8 Variablen	16
2.9 Datentypen und Klassen	18
2.10 Arithmetische Zuweisungsoperatoren	28
2.11 Mehrfache Zuweisungen (Unpacking)	28
2.12 Indizierung von Ausschnitten (Slices)	29
2.13 Formatierte Strings	30
2.14 Vergleiche und boolesche Ausdrücke	31
2.15 Erstellen von Python-Programmen	33
2.16 Kontrollstrukturen	35
2.17 Funktionen	36
2.18 Funktionen mit optionalen Argumenten	38
2.19 Bedingte Ausführung von Anweisungen	39
2.20 Bedingte Wiederholung von Anweisungen	39
2.21 Schleifen über eine Aufzählung von Elementen	41
2.22 Schleifen mit zip und enumerate	43
2.23 Styleguide PEP 8	44
Zusammenfassung	46
Aufgaben	46
Literatur	48
3 NumPy und Matplotlib	49
3.1 Eindimensionale Arrays	50
3.2 Mehrdimensionale Arrays	51
3.3 Datentypen in NumPy	52
3.4 Rechnen mit Arrays	54
3.5 Erzeugen von Arrays	55

3.6	Indizierung von Array-Ausschnitten (Array-Slices)	56
3.7	Indizierung mit ganzzahligen Arrays	58
3.8	Indizierung mit booleschen Arrays	58
3.9	Ausgelassene Indizes	59
3.10	Logische Operationen auf Arrays	60
3.11	Mehrfache Zuweisungen mit Arrays (Unpacking)	61
3.12	Broadcasting	62
3.13	Matrixmultiplikationen mit @	63
3.14	Lösen von linearen Gleichungssystemen	65
3.15	Änderung der Form von Arrays	66
3.16	Grafische Ausgaben mit Matplotlib	67
3.17	Animationen mit Matplotlib	70
3.18	Positionierung von Grafikelementen	74
	Zusammenfassung	77
	Aufgaben	78
	Literatur	80
4	Physikalische Größen und Messungen	81
4.1	Darstellung physikalischer Größen	81
4.2	Statistische Messfehler	83
4.3	Simulation der Gauß-Verteilung	90
4.4	Grafische Darstellung von Messdaten	92
4.5	Kurvenanpassung an Messdaten	94
	Zusammenfassung	96
	Aufgaben	97
	Literatur	98
5	Kinematik des Massenpunkts	99
5.1	Schiefer Wurf	100
5.2	Radiodromen	103
5.3	Gleichförmige Kreisbewegung	112
5.4	Bewegung entlang einer Schraubenlinie	115
	Zusammenfassung	117
	Aufgaben	117
	Literatur	120
6	Statik von Massenpunkten	121
6.1	Starre Stabwerke	123
6.2	Elastische Stabwerke	132
6.3	Linearisierung kleiner Deformationen	140
6.4	Darstellung der Kräfte über eine Farbtabelle	146
6.5	Dreidimensionale Stabwerke	148
6.6	Unterbestimmte Stabwerke	148
	Zusammenfassung	148
	Aufgaben	149
	Literatur	150

7	Dynamik des Massenpunkts	151
7.1	Eindimensionale Bewegungen	152
7.2	Reduktion der Ordnung	155
7.3	Runge-Kutta-Verfahren	156
7.4	Freier Fall mit Luftreibung	160
7.5	Interpolation von Messwerten für Simulationen	163
7.6	Mehrdimensionale Bewegungen	166
7.7	Schiefer Wurf mit Luftreibung	167
7.8	Schiefer Wurf mit Coriolis-Kraft	170
7.9	Planetenbahnen	173
	Zusammenfassung	177
	Aufgaben	179
	Literatur	180
8	Mehrteilchensysteme und Erhaltungssätze	181
8.1	Erhaltungssätze	181
8.2	Bewegungen mehrerer Massen	183
8.3	Doppelsternsysteme	184
8.4	Sonnensystem	189
8.5	Elastische Stoßprozesse	195
8.6	Stoß zweier harter Kugeln	199
8.7	Stoß vieler harter Kugeln	206
8.8	Modell eines Gases	217
8.9	Gleichverteilungssatz der statistischen Physik	219
8.10	Brownsche Bewegung	221
	Zusammenfassung	222
	Aufgaben	223
	Literatur	224
9	Zwangsbedingungen	225
9.1	Verallgemeinerte Koordinaten	226
9.2	Pendel mit Zwangskraft	228
9.3	Baumgarte-Stabilisierung	232
9.4	Verallgemeinerung der Zwangsbedingungen	235
9.5	Chaotische Mehrfachpendel	240
9.6	Zwangsbedingungen mit Ungleichungen	247
9.7	Zeitabhängige Zwangsbedingungen	257
	Zusammenfassung	258
	Aufgaben	259
	Literatur	262
10	Schwingungen	263
10.1	Theorie des linearen Federpendels	264
10.2	Darstellung von Resonanzkurven	265
10.3	Visualisierung einer Feder	267
10.4	Simulation des Federpendels	271
10.5	Hörbarmachen von Schwingungen	275

10.6	Nichtlineare Schwingungen	278
10.7	Fourier-Analysen	281
10.8	Spektralanalyse von Audiosignalen	287
10.9	Amplituden und Frequenzmodulation	291
10.10	Resonanzkurven nichtlinearer Systeme	294
10.11	Gekoppelte Schwingungen und Eigenmoden	298
	Zusammenfassung	307
	Aufgaben	308
	Literatur	310
11	Wellen	311
11.1	Transversal- und Longitudinalwellen	313
11.2	Masse-Feder-Kette	316
11.3	Stehende Wellen	321
11.4	Interferenz	323
11.5	Komplexe Amplituden	326
11.6	Huygenssches Prinzip: Beugung am Spalt	328
11.7	Brechung	331
11.8	Doppler-Effekt und machscher Kegel	335
11.9	Hörbarmachen des Doppler-Effekts	338
11.10	Dispersion und Gruppengeschwindigkeit	343
11.11	Zerfließen eines Wellenpakets	346
	Zusammenfassung	349
	Aufgaben	350
	Literatur	352
12	Grafische Benutzeroberflächen	353
12.1	Objektorientierte Programmierung	354
12.2	Definition von Klassen	356
12.3	Vererbung	358
12.4	Überschreiben von Methoden	359
12.5	Erzeugen einer Benutzeroberfläche mit PyQt	360
12.6	Design einer Benutzeroberfläche	361
12.7	Implementierung der Benutzeroberfläche	363
12.8	Direkte Verwendung einer .ui-Datei mit PyQt	366
12.9	Vorteile von pyuic gegenüber loadUiType	366
12.10	Benutzeroberfläche für den schiefen Wurf	367
12.11	Implementierung der Benutzeroberfläche	370
12.12	Generatoren	376
12.13	Animationen in GUIs	377
	Zusammenfassung	379
	Literatur	380
13	Objektorientierte Simulationen	381
13.1	Dekoratoren	382
13.2	Properties	384
13.3	Entpacken von Funktionsargumenten	385

13.4	Funktionen mit variabler Anzahl von Argumenten	386
13.5	Problemanalyse der Stabwerke	387
13.6	Das Paket Stabwerke	389
13.7	Die Klasse Stabwerk	390
13.8	Die Klasse StabwerkStarr	395
13.9	Die Klasse StabwerkElastisch	396
13.10	Die Klasse StabwerkElastischLin	398
13.11	Die Klasse PlotStabwerk	403
13.12	Die Klasse AnimationEigenmode	404
13.13	Anwendungen des Pakets	405
	Zusammenfassung	410
	Aufgaben	410
	Literatur	412
14	Ausblick	413
	Literatur	414
	Index	415